# Anomaly Detection On Graph

Linghao Chen

# SpotLight: Detecting Anomalies in Streaming Graphs On Streams

## SpotLight: Detecting Anomalies in Streaming Graphs


Dhivya Eswaran*
Christos Faloutsos*
{deswaran,christos}@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Sudipto Guha
sudipto@amazon.com
Amazon
New York City, NY, USA

Nina Mishra
nmishra@amazon.com
Amazon
Palo Alto, CA, USA


KDD-18

# SpotLight: Detecting Anomalies in Streaming Graphs On Streams

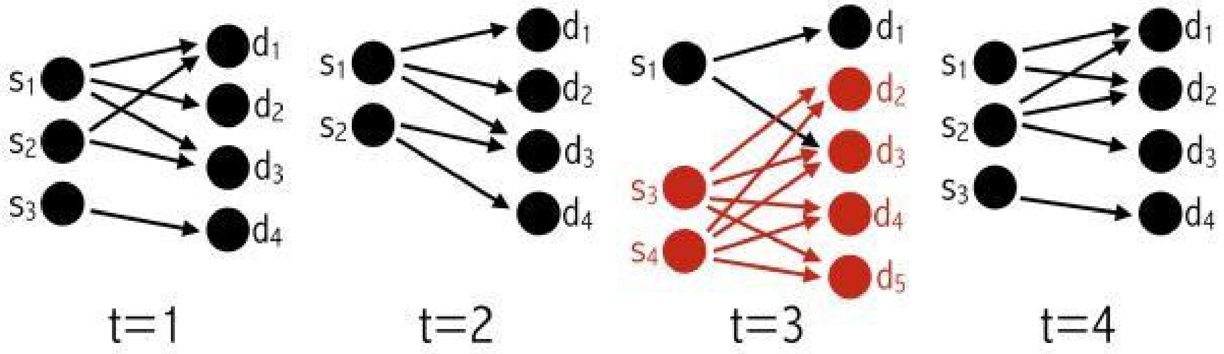



**Figure 1: Sudden appearance of a dense subgraph at $t$=3.**

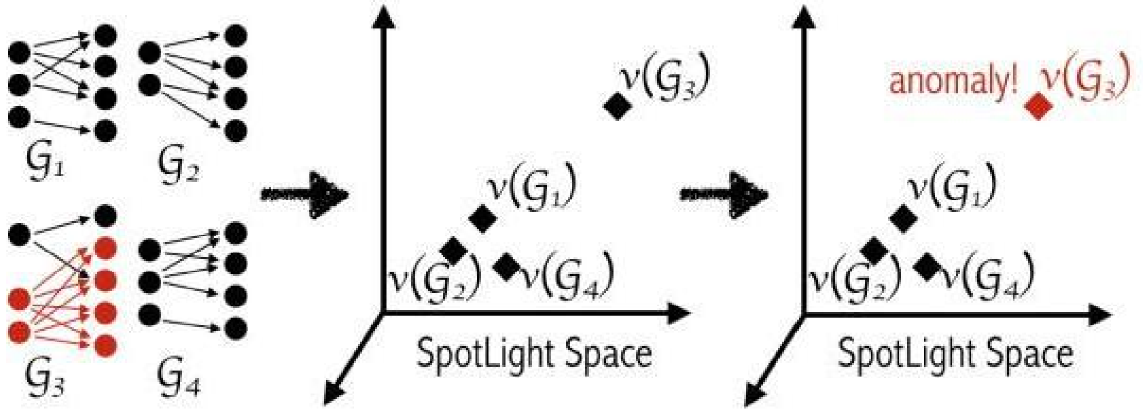



**Figure 2: Overview of SpotLight**

# SpotLight: Detecting Anomalies in Streaming Graphs On Streams

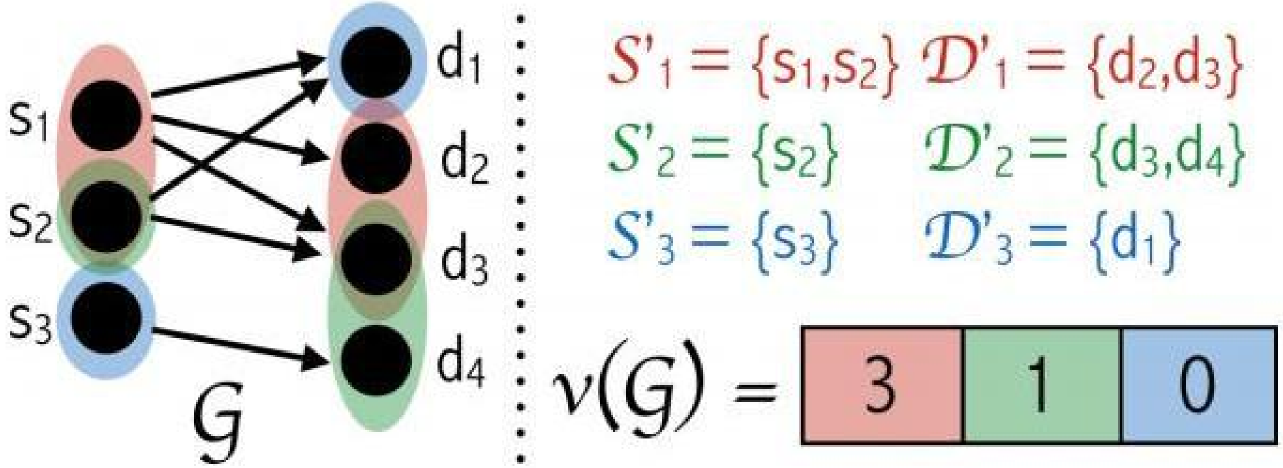



**Figure 3: A** $(K{=}3, p{=}0.5, q{=}0.33)$**-SpotLight sketch** $\mathrm{v}(\mathcal{G})$ **of a graph** $\mathcal{G}$ **with unit-weight edges. Each sketch dimension** $v_k(\mathcal{G})$ **is the total weight of edges going from a random set of sources** $\mathcal{S}'_k$ **and to a random set of destinations** $\mathcal{D}'_k$**.**

# SpotLight: Detecting Anomalies in Streaming Graphs On Streams

'normal' instances in the SpotLight (sketch) space, we may now employ any off-the-shelf data stream anomaly detector (e.g., [9, 20, 29]) to carry out AnomalyScore procedure call (line 5 of Alg. 1). These

---

## Robust Random Cut Forest Based Anomaly Detection On Streams

---


**Sudipto Guha** SUDIPTO@CIS.UPENN.EDU
University of Pennsylvania, Philadelphia, PA 19104.

**Nina Mishra** NMISHRA@AMAZON.COM
Amazon, Palo Alto, CA 94303.

**Gourav Roy** GOURAVR@AMAZON.COM
Amazon, Bangalore, India 560055.

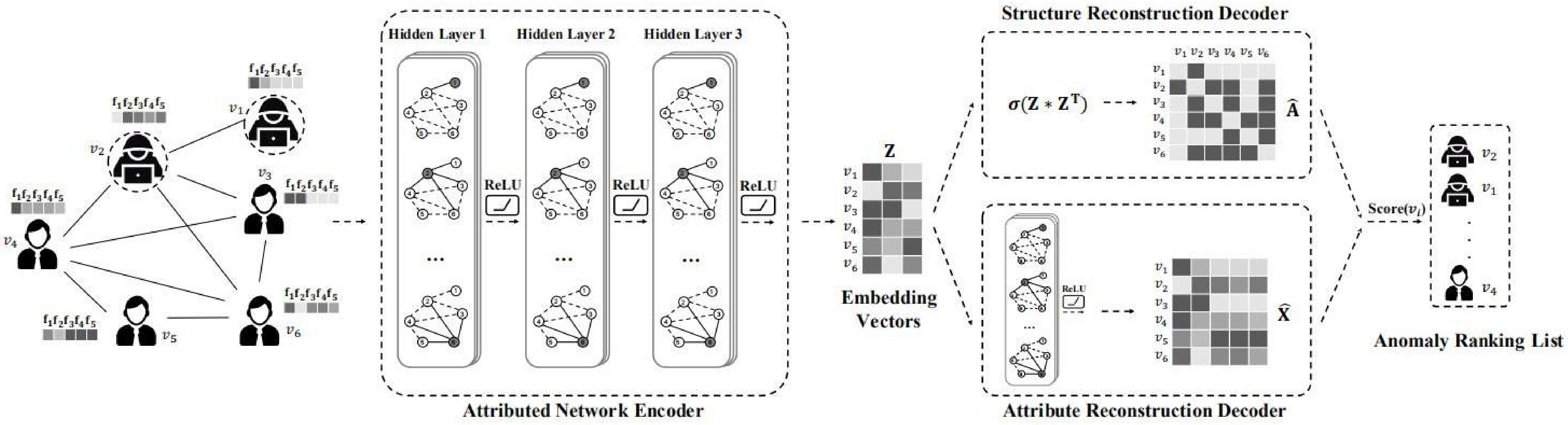
**Okke Schrijvers** OKKES@CS.STANFORD.EDU
Stanford University, Palo Alto, CA 94305.

# Deep Anomaly Detection on Attributed Networks

Kaize Ding* Jundong Li* Rohit Bhanushali* Huan Liu*

ICDM '19

# Deep Anomaly Detection on Attributed Networks



# Deep Anomaly Detection on Attributed Networks

## Attributed Network Encoder

$$\mathbf{H}^{(l+1)} = f(\mathbf{H}^{(l)}, \mathbf{A}|\mathbf{W}^{(l)}) = \sigma(\widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$$

$$\mathbf{H}^{(1)} = f_{Relu}(\mathbf{X}, \mathbf{A}|\mathbf{W}^{(0)})$$
$$\mathbf{H}^{(2)} = f_{Relu}(\mathbf{H}^{(1)}, \mathbf{A}|\mathbf{W}^{(1)})$$
$$\mathbf{Z} = \mathbf{H}^{(3)} = f_{Relu}(\mathbf{H}^{(2)}, \mathbf{A}|\mathbf{W}^{(2)})$$

## Structure Reconstruction

$$p(\widehat{\mathbf{A}}_{i,j} = 1|\mathbf{z}_i, \mathbf{z}_j) = sigmoid(\mathbf{z}_i, \mathbf{z}_j^{\mathrm{T}})$$
$$\widehat{\mathbf{A}} = sigmoid(\mathbf{Z}\mathbf{Z}^{\mathrm{T}})$$

## Attribute Reconstruction Decoder

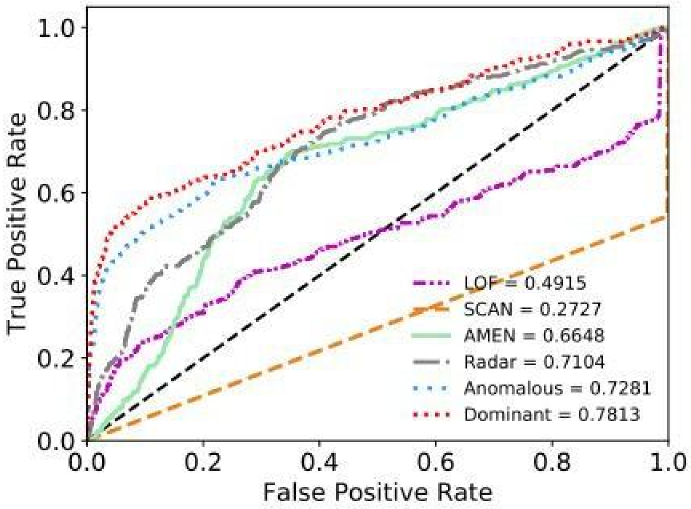
$$\widehat{\mathbf{X}} = f_{Relu}(\mathbf{Z}, \mathbf{A}|\mathbf{W}^{(3)})$$

# Deep Anomaly Detection on Attributed Networks

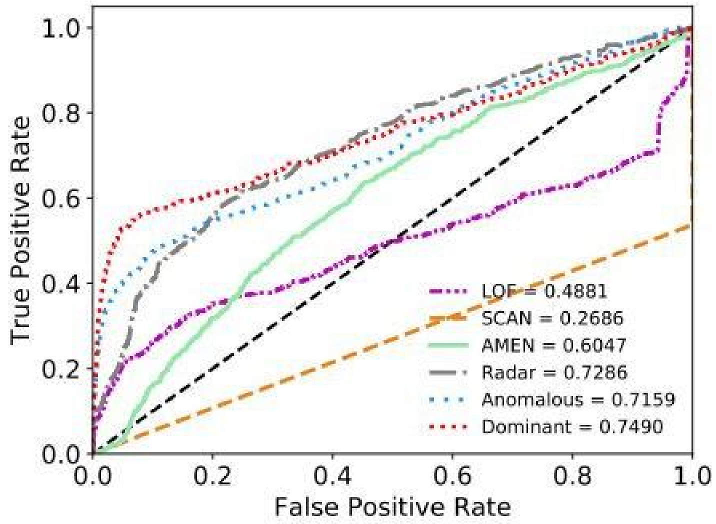
$$\begin{aligned}\mathcal{L} &= (1-\alpha)\mathbf{R}_S + \alpha\mathbf{R}_A \\ &= (1-\alpha)||\mathbf{A} - \widehat{\mathbf{A}}||_F^2, + \alpha||\mathbf{X} - \widehat{\mathbf{X}}||_F^2\end{aligned}$$

$$score(\mathbf{v}_i) = (1-\alpha)||\mathbf{a} - \widehat{\mathbf{a}}_i||_2 + \alpha||\mathbf{x}_i - \widehat{\mathbf{x}}_i||_2.$$

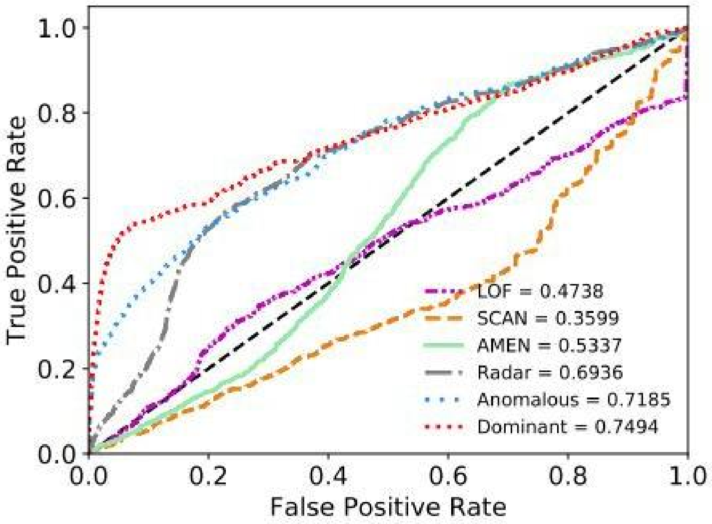
**ROC-AUC:**




(a) BlogCatalog




(b) Flickr




(c) ACM

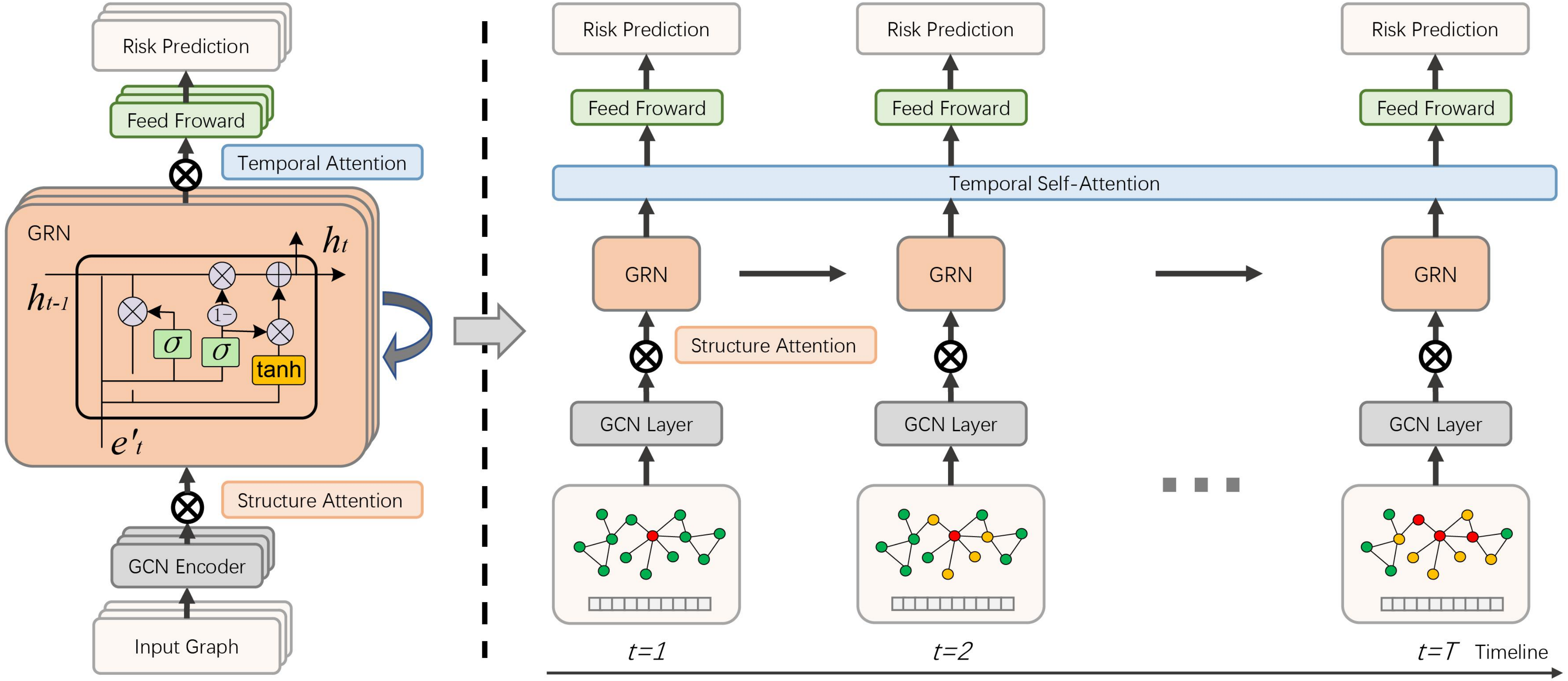
# Risk Guarantee Prediction in Networked-Loans

**Dawei Cheng**[1], **Xiaoyang Wang**[2], **Ying Zhang**[3] and **Liqing Zhang**[1*]
[1]MoE Key Lab of Artificial Intelligence, Department of Computer Science and Engineering,
Shanghai Jiao Tong University, China
[2]Zhejiang Gongshang University, China
[3]University of Technology Sydney, Australia
dawei.cheng@sjtu.edu.cn, xiaoyangw@zjgsu.edu.cn, ying.zhang@uts.edu.au, zhang-lq@cs.sjtu.edu.cn

IJCAI-20

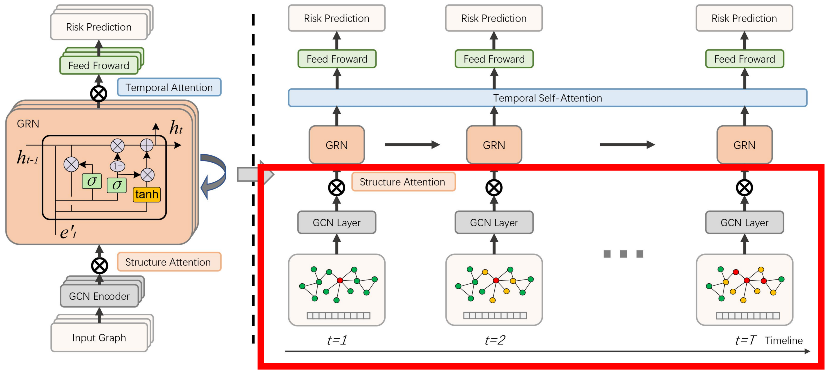
# Risk Guarantee Prediction in Networked-Loans

应用：信贷网络edge的的risk预测，检测的是边。



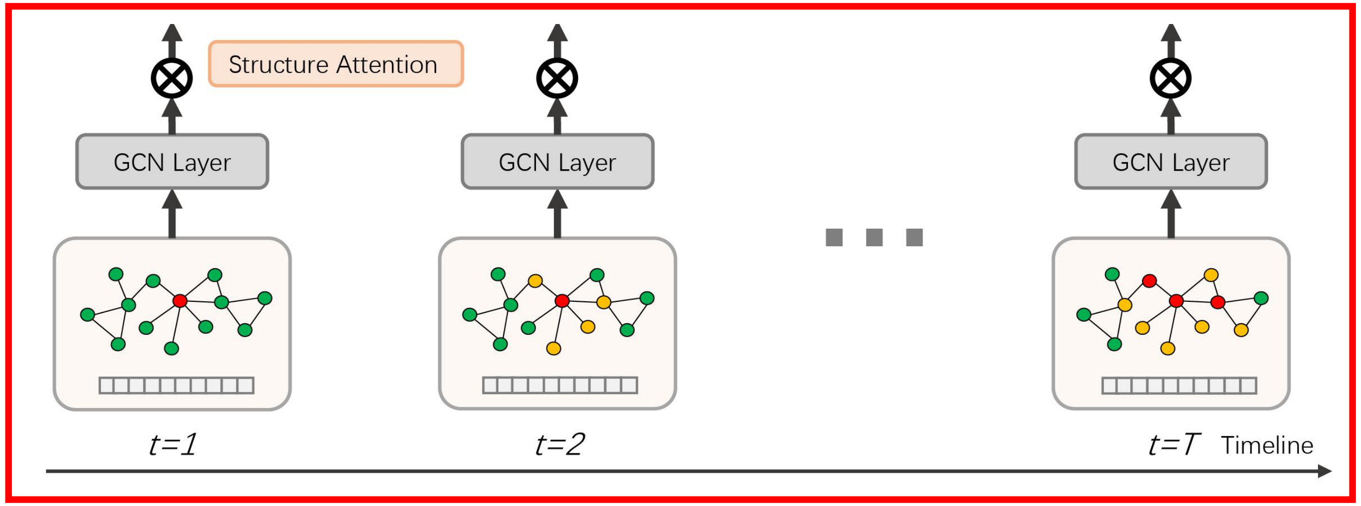
# Risk Guarantee Prediction in Networked-Loans

每个时间节点上的graph先Conv，Conv之后用Attention更新节点信息。

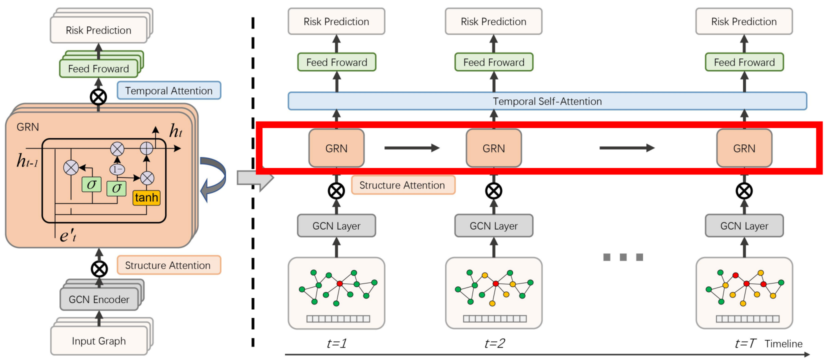



$$\alpha_{i,j} = \frac{\exp(\mathrm{Conv}(W_c v_i, W_c v_j))}{\sum_{k \in \mathcal{N}_i} \exp(\mathrm{Conv}(W_c v_i, W_c v_k))}$$

$$v_i' = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in \mathcal{N}_i} \alpha_{i,j}^{k} \mathrm{Conv}(W_c^k v_j)\right)$$



# Risk Guarantee Prediction in Networked-Loans



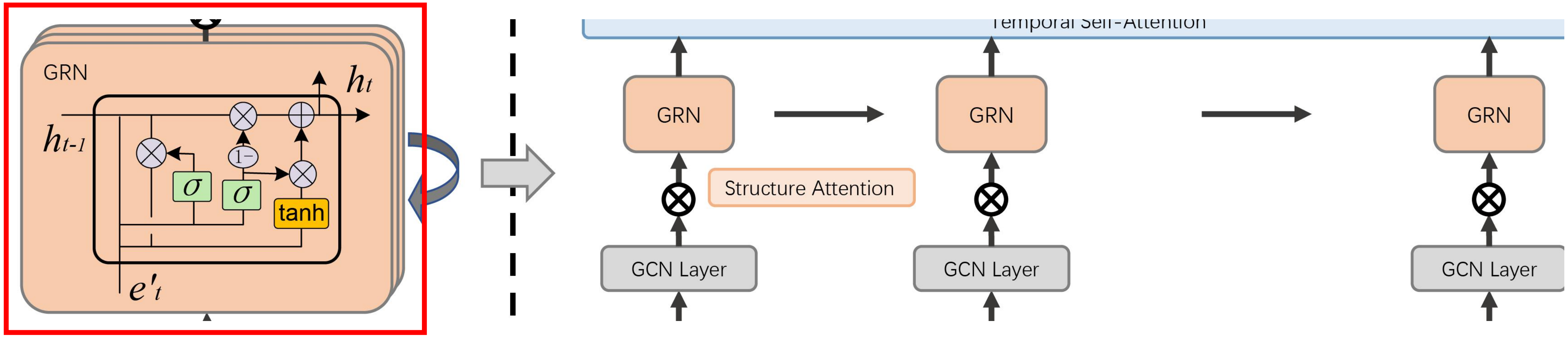

Attention之后的结点，通过$e_{ij} = \{v_i, v_j, m_{ij}\}$， $m_{ij}$为借贷金额。将边信息投入GRN时序网络。

$$z_t = \sigma(W_z[h_{t-1}, e_t'])$$

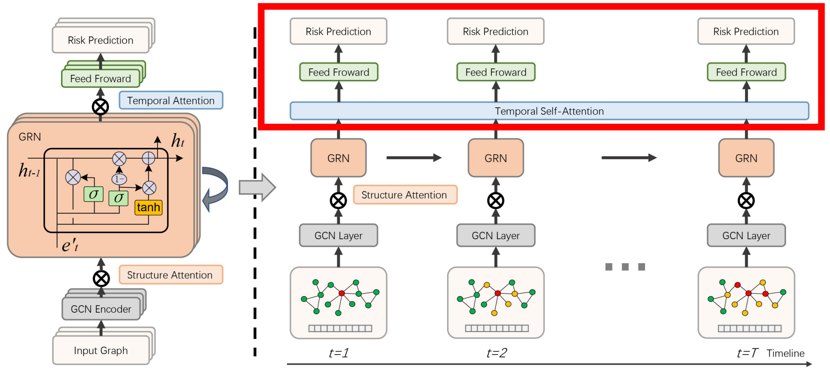
$$r_t = \sigma(W_r[h_{t-1}, e_t'])$$

$$\widetilde{h}_t = \tanh(W_h[r_t * h_{t-1}, e_t'])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h}_t$$



# Risk Guarantee Prediction in Networked-Loans




对edge做不同时序之间的attention，更新edge的信息

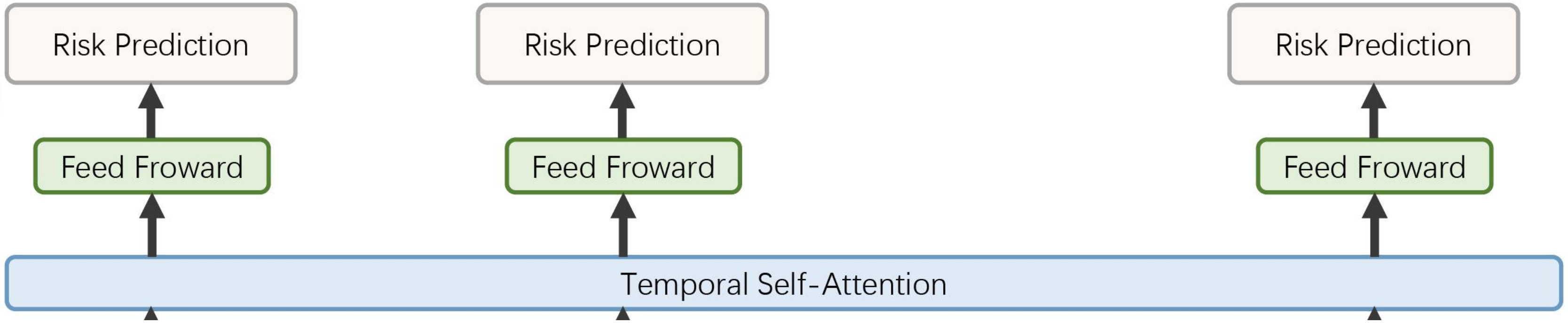
$$u_t = (\|_{j=1}^{T} \beta_{t,j} W_t h_j)$$

$$\beta_{t,j} = \frac{\exp(\mathrm{NN}(W_n h_t, W_n h_j))}{\sum_{i=1}^{T} \exp(\mathrm{NN}(W_n h_t, W_n h_i))}$$

NN denotes a feed forward neural network layer with ReLU as activation function.

训练分类器，做2-classification

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\mathrm{pred}(u_i : \theta)) + \lambda(1 - y_i)\log(1 - \mathrm{pred}(u_i : \theta))]$$



# Inductive Anomaly Detection on Attributed Networks

**Kaize Ding**[1] , **Jundong Li**[2,3] , **Nitin Agarwal**[4] and **Huan Liu**[1]
[1]Computer Science and Engineering, Arizona State University, USA
[2]Electrical and Computer Engineering, University of Virginia, USA
[3]Computer Science & School of Data Science, University of Virginia, USA
[4]Information Science, University of Arkansas Little Rock, USA
kaize.ding@asu.edu, jundong@virginia.edu, nxagarwal@ualr.edu, huan.liu@asu.edu

**IJCAI-20**

## An unsupervised framework

**AEGIS** trains a generative adversarial network (Ano-GAN) to improve the model generalization ability on newly added data. Specifically, the **generator aims to generate informative potential anomalies**, while the **discriminator tries to learn a decision boundary that separates the potential anomalies from the normal data**.

用GAN生成异常数据

# Inductive Anomaly Detection on Attributed Networks

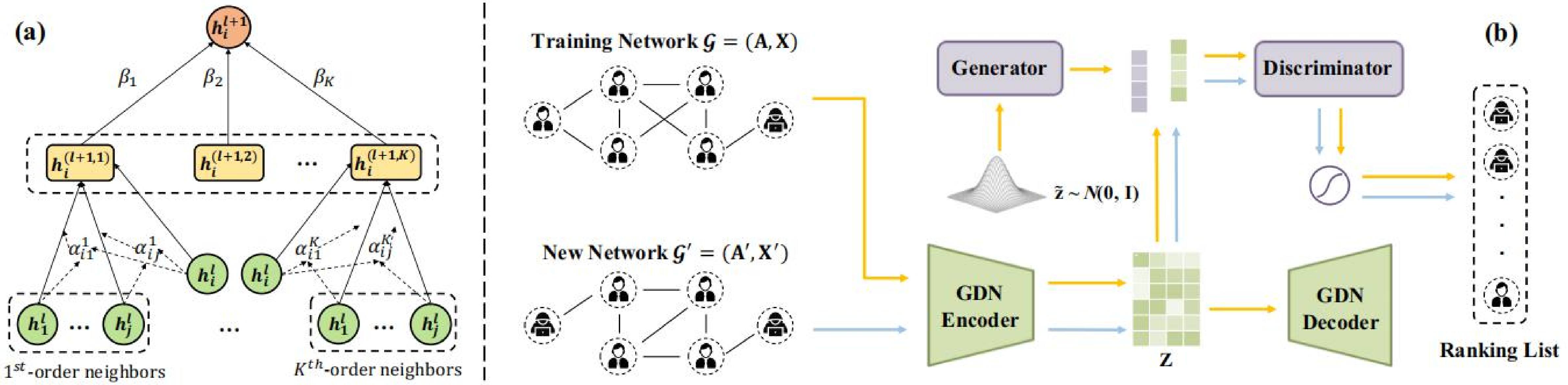
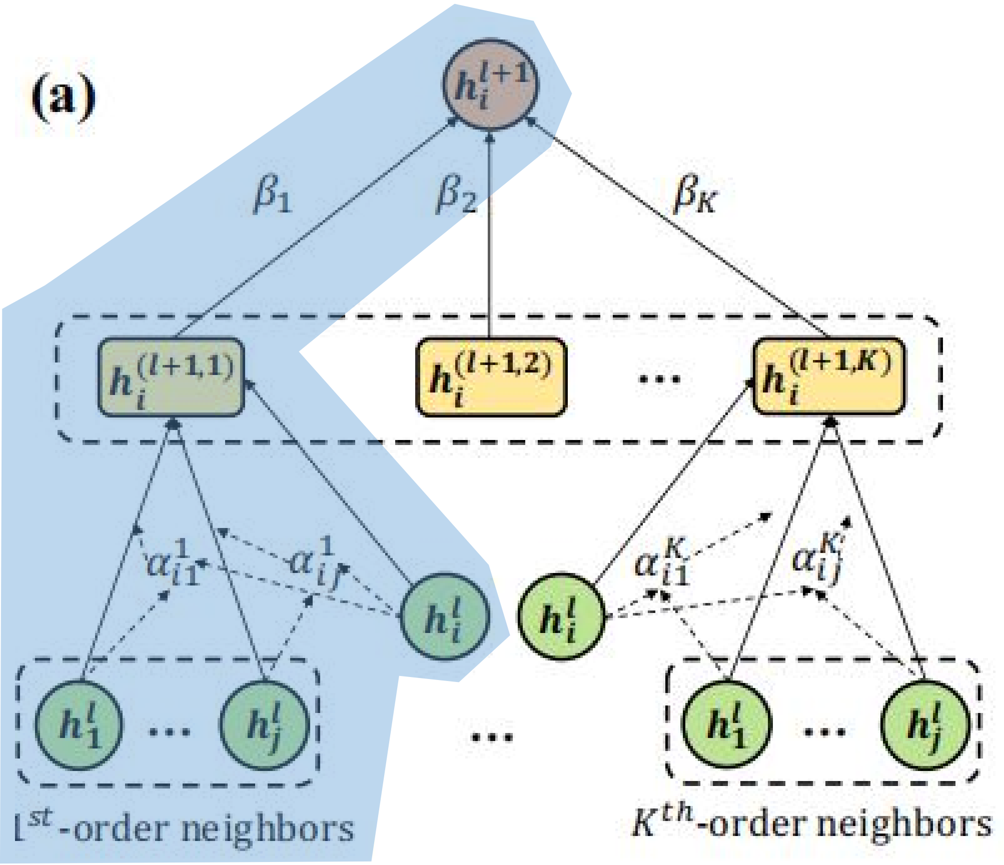



Figure 1: (a) The graph differentiative layer. (b) The proposed inductive anomaly detection framework AEGIS. Note that AEGIS is trained with the partially observed network $\mathcal{G}$, and can directly detect anomalies on the new network $\mathcal{G}'$ in a feed-forward way. The yellow arrows denote the training flow and the blue arrows denote the inference flow. Figure best viewed in color.

## Graph Differentiative Layer : GDN




一阶邻居

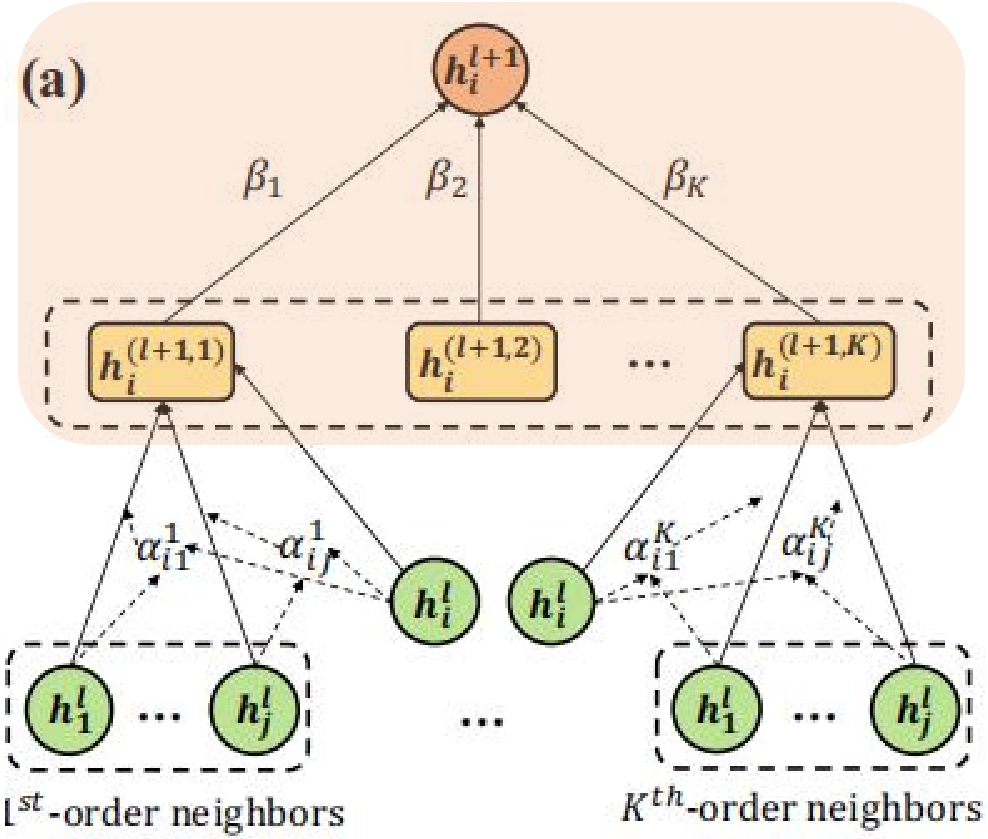
一阶邻居:

$$\mathbf{h}_i^{(l)} = \sigma\left(\mathbf{W}_1\mathbf{h}_i^{(l-1)} + \sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}_2\Delta_{i,j}^{(l-1)}\right), \qquad (1)$$

where $\mathbf{h}^{(l-1)} \in \mathbb{R}^F, \mathbf{h}^{(l)} \in \mathbb{R}^{\tilde{F}}$ denotes the input and output representation of node $i$, respectively. $\Delta_{i,j}^{(l-1)} = \mathbf{h}_i^{(l-1)} - \mathbf{h}_j^{(l-1)}$ is the feature difference between node $i$ and $j$. $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{F\times\tilde{F}}$ are two trainable weight matrices and $\sigma$ is a nonlinear activation function. $\mathcal{N}_i$ denotes the neighboring nodes of node $i$. Here $\alpha_{ij}$ is the attention coefficient between node $i$ and node $j$, which can be expressed as:

$$\alpha_{ij} = \frac{\exp(\sigma(\mathbf{a}^{\mathrm{T}}\mathbf{W}_2\Delta_{i,j}^{(l-1)}))}{\sum_{k\in\mathcal{N}_i}\exp(\sigma(\mathbf{a}^{\mathrm{T}}\mathbf{W}_2\Delta_{i,k}^{(l-1)}))}, \qquad (2)$$

# Graph Differentiative Layer: GDN




一阶邻居

K阶邻居

**找到K阶邻居:**

Similarly, by extracting $k^{th}$-order neighbors of node $i$ from $\mathbf{A}^k = \underbrace{\mathbf{A} \cdot \mathbf{A} \ldots \mathbf{A}}_{k}$, we can compute its $k^{th}$-order node representation $\mathbf{h}_i^{(l,k)}$. As different neighborhoods encode differ-

**聚合:**

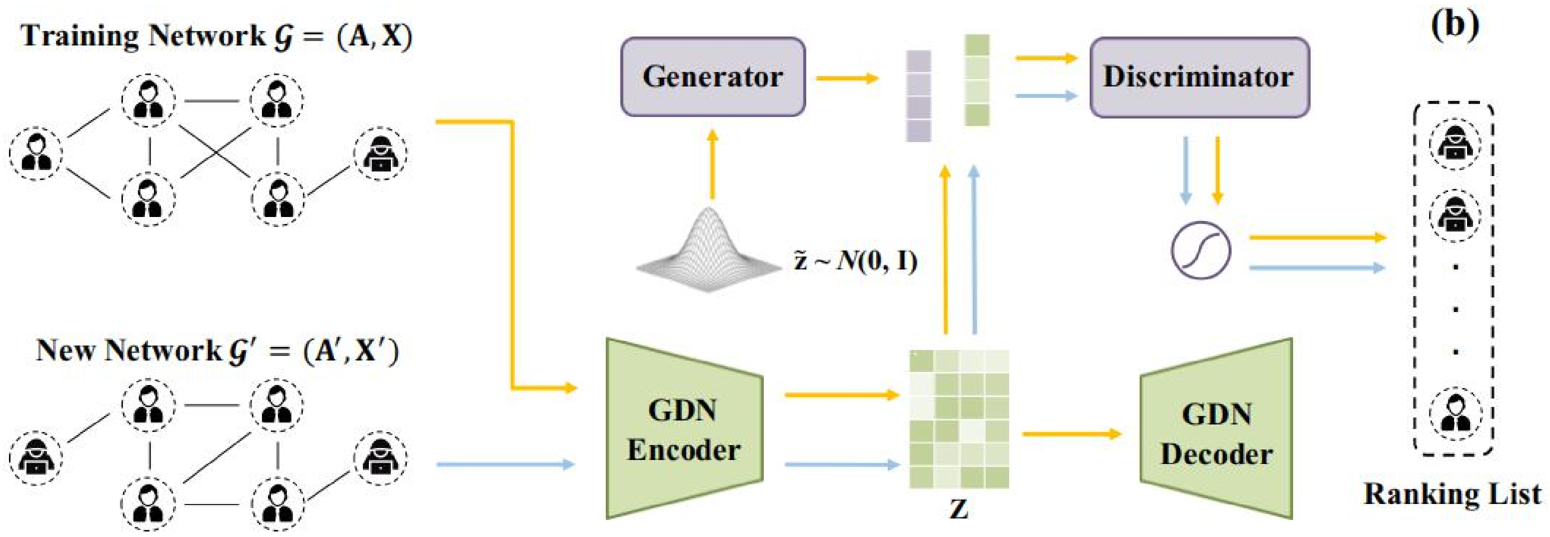
$$\mathbf{h}_i^{(l)} = \sum_{k=1}^{K} \beta_i^k \mathbf{h}_i^{(l,k)}, \tag{3}$$

where $\beta_i^k$ denotes the attention coefficient on $k^{th}$-order representation $\mathbf{h}_i^{(l,k)}$, which can be formulated as:

$$\beta_i^k = \frac{\exp\left(\sigma(\hat{\mathbf{a}}^{\mathrm{T}} \mathbf{h}_i^{(l,k)})\right)}{\sum_{k'=1}^{K} \exp\left(\sigma(\hat{\mathbf{a}}^{\mathrm{T}} \mathbf{h}_i^{(l,k')})\right)}. \tag{4}$$

得到了结点的表示

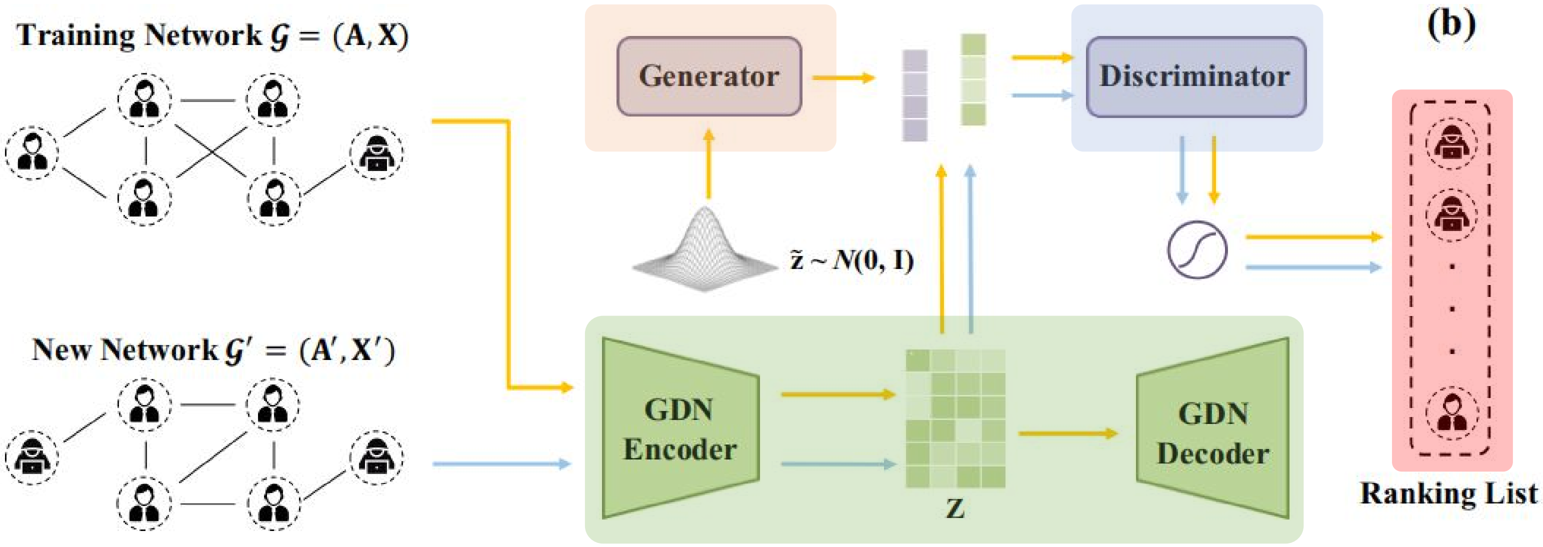
# Adversarial Graph Differentiation Networks




The yellow arrows denote the training flow and the blue arrows denote the inference flflow. Figure best viewed in color.

The generator *G* effectively improves the capability of the discriminator *D* to identify normal data by generating informative potential anomalies.

# Adversarial Graph Differentiation Networks




$$\mathcal{L}_{\mathrm{AE}} = \frac{1}{n}\sum_{i=1}^{n} ||Dec(Enc(\mathbf{x}_i)) - \mathbf{x}_i||^2$$

For the generator $G$, the loss function can be defifined as:

$$\mathcal{L}_{\mathrm{G}} = \mathbb{E}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}})}[\log(1 - D(G(\tilde{\mathbf{z}})))]$$

and the loss function of learning discriminator $D$ is:

$$\mathcal{L}_{\mathrm{D}} = -\mathbb{E}_{\mathbf{z}\sim\mathbf{Z}}[\log D(\mathbf{z})] - \mathbb{E}_{\tilde{\mathbf{z}}\sim p(\tilde{\mathbf{z}})}[\log(1 - D(G(\tilde{\mathbf{z}})))]$$

**Anomaly score:** $score(\mathbf{x}'_i) = p(y'_i = 0|\mathbf{z}'_i) = 1 - D(\mathbf{z}'_i).$

# DEEP GRAPH INFOMAX

**Petar Veličković***
Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

**William Fedus**
Mila – Québec Artificial Intelligence Institute
Google Brain
liamfedus@google.com

**William L. Hamilton**
Mila – Québec Artificial Intelligence Institute
McGill University
wlh@cs.mcgill.ca

**Pietro Liò**
Department of Computer Science and Technology
University of Cambridge
pietro.lio@cst.cam.ac.uk

**Yoshua Bengio**[†]
Mila – Québec Artificial Intelligence Institute
Université de Montréal
yoshua.bengio@mila.quebec

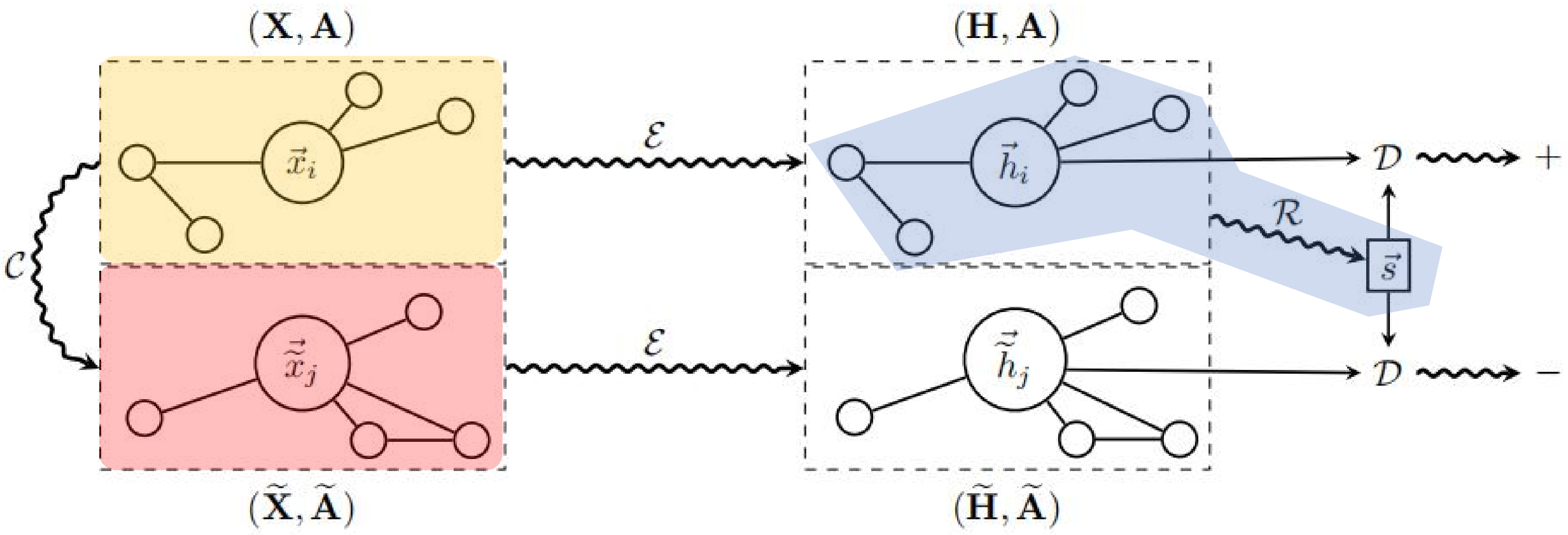
**R Devon Hjelm**
Microsoft Research
Mila – Québec Artificial Intelligence Institute
devon.hjelm@microsoft.com

**ICLR-19**

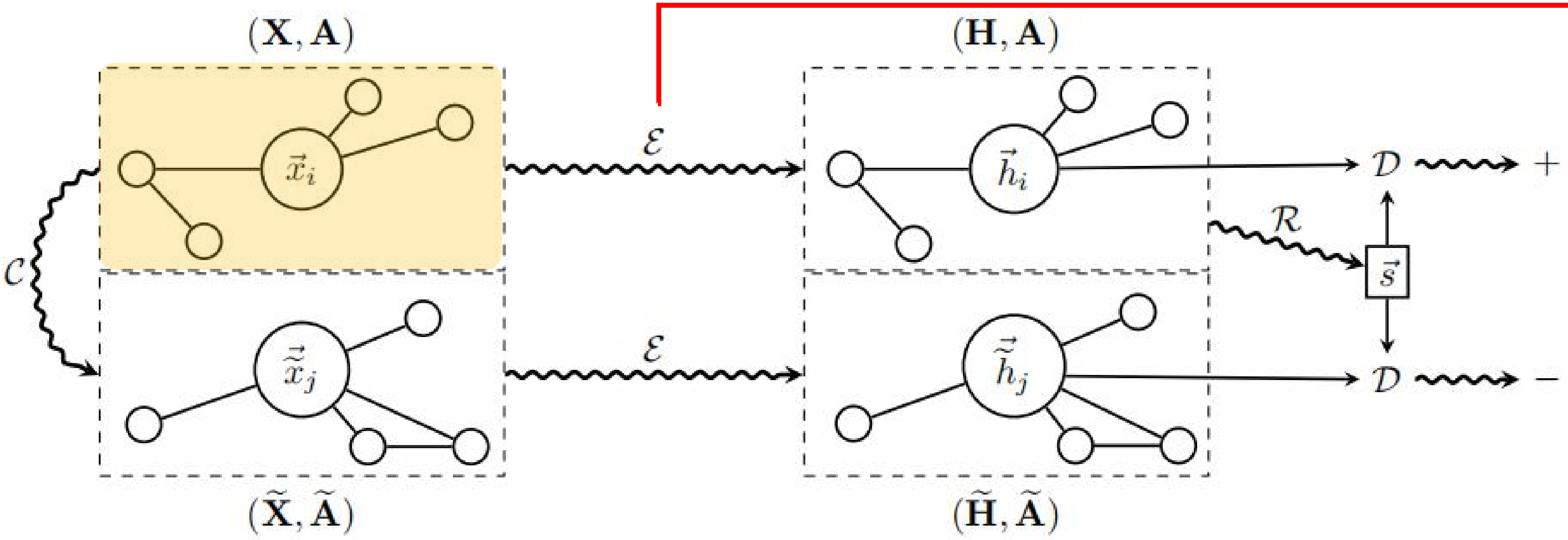
下面一篇中文论文用到的方法

# DEEP GRAPH INFOMAX

1. Sample a negative example by using the corruption function: $(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}}) \sim \mathcal{C}(\mathbf{X}, \mathbf{A})$.
2. Obtain patch representations, $\vec{h}_i$ for the input graph by passing it through the encoder: $\mathbf{H} = \mathcal{E}(\mathbf{X}, \mathbf{A}) = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}$.
3. Obtain patch representations, $\vec{\widetilde{h}}_j$ for the negative example by passing it through the encoder: $\widetilde{\mathbf{H}} = \mathcal{E}(\widetilde{\mathbf{X}}, \widetilde{\mathbf{A}}) = \{\vec{\widetilde{h}}_1, \vec{\widetilde{h}}_2, \ldots, \vec{\widetilde{h}}_M\}$.
4. Summarize the input graph by passing its patch representations through the readout function: $\vec{s} = \mathcal{R}(\mathbf{H})$.
5. Update parameters of $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{D}$ by applying gradient descent to maximize Equation 1.



## Transductive learning



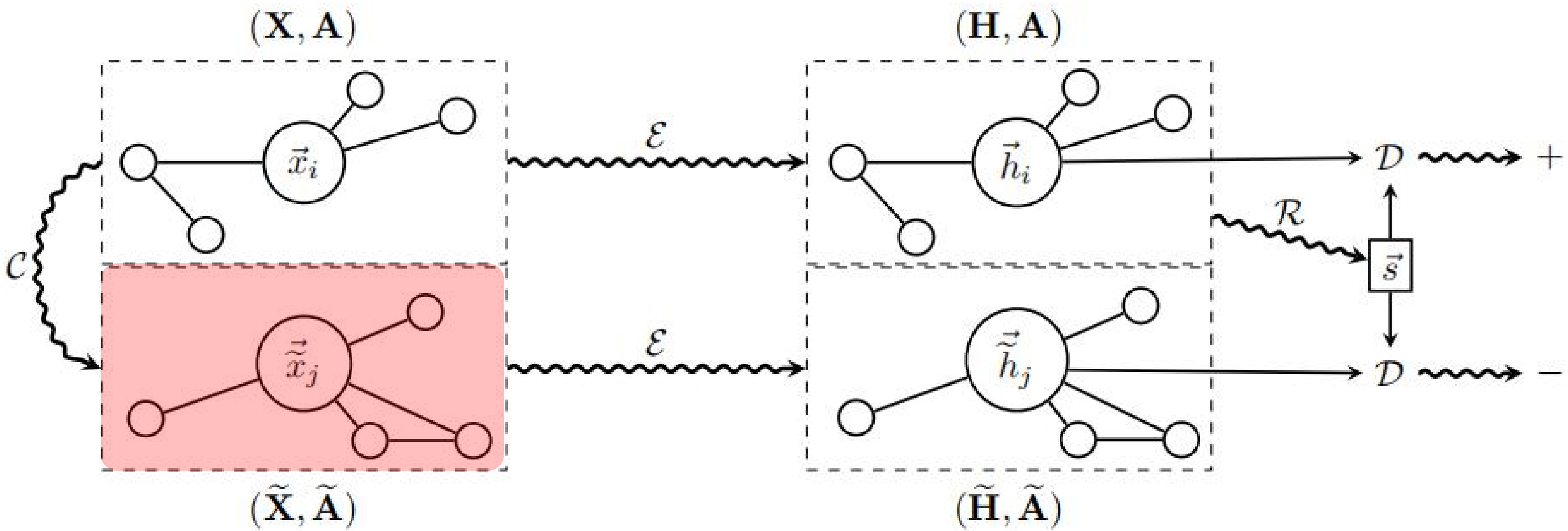

$$\mathcal{E}(\mathbf{X}, \mathbf{A}) = \sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{\Theta}\right)$$

## Inductive learning on large graphs

**GraphSAGE-GCN** $\mathrm{MP}(\mathbf{X}, \mathbf{A}) = \hat{\mathbf{D}}^{-1}\hat{\mathbf{A}}\mathbf{X}\mathbf{\Theta}$ $\quad \widetilde{\mathrm{MP}}(\mathbf{X}, \mathbf{A}) = \sigma\left(\mathbf{X}\mathbf{\Theta}' \| \mathrm{MP}(\mathbf{X}, \mathbf{A})\right)$ $\quad \mathcal{E}(\mathbf{X}, \mathbf{A}) = \widetilde{\mathrm{MP}}_3(\widetilde{\mathrm{MP}}_2(\widetilde{\mathrm{MP}}_1(\mathbf{X}, \mathbf{A}), \mathbf{A}), \mathbf{A})$

$$\begin{aligned}
\mathbf{H}_1 &= \sigma\left(\mathrm{MP}_1(\mathbf{X}, \mathbf{A})\right) \\
\mathbf{H}_2 &= \sigma\left(\mathrm{MP}_2(\mathbf{H}_1 + \mathbf{X}\mathbf{W}_{\text{skip}}, \mathbf{A})\right) \\
\mathcal{E}(\mathbf{X}, \mathbf{A}) &= \sigma\left(\mathrm{MP}_3(\mathbf{H}_2 + \mathbf{H}_1 + \mathbf{X}\mathbf{W}_{\text{skip}}, \mathbf{A})\right)
\end{aligned}$$

## Corruption function



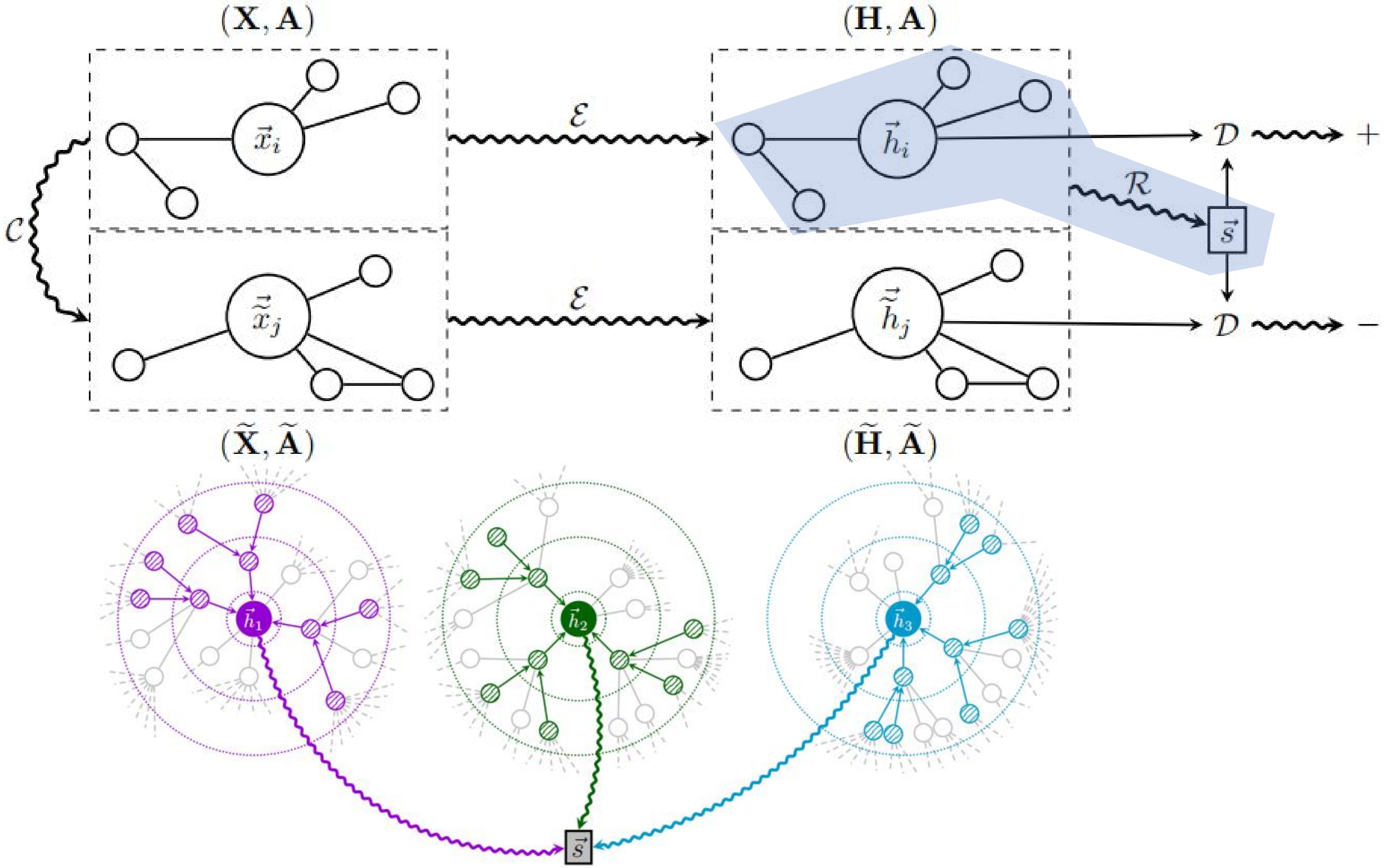

$C$ **preserves** the original adjacency matrix ($\tilde{A} = A$), whereas the corrupted features, $\widetilde{X}$, are obtained by **row-wise shufflfling** of $X$.

**the same nodes** as the original graph

located in **different places** in the graph

DGI is **stable** to other choices of corruption functions!!!

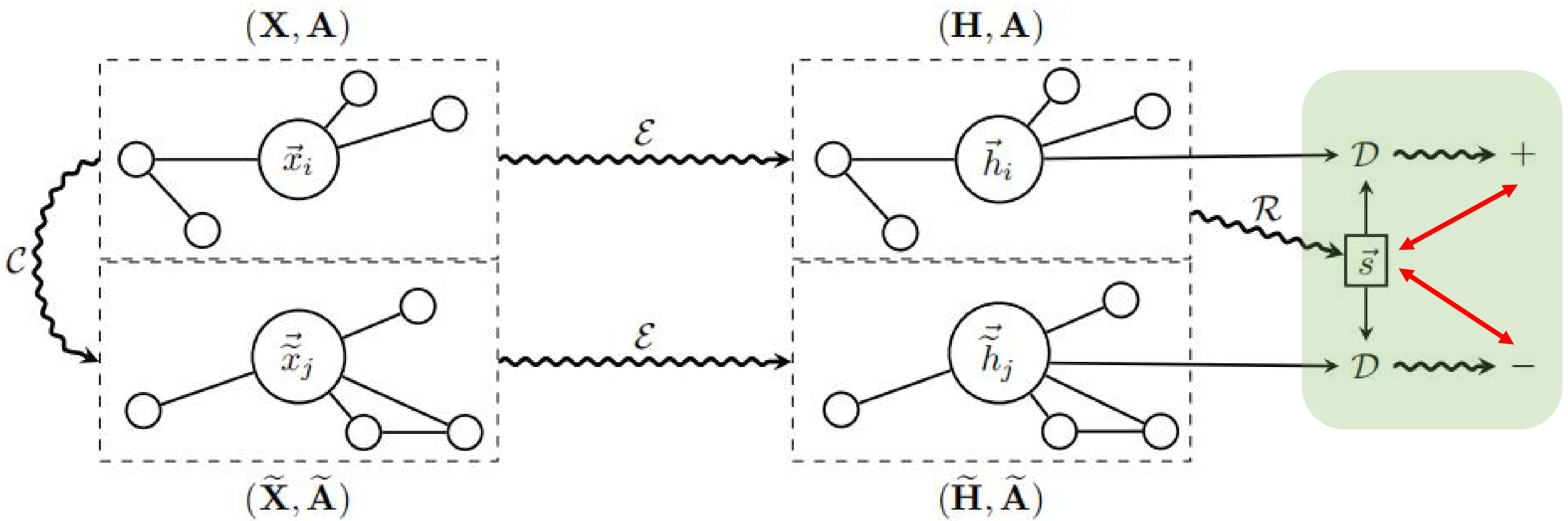
# Training details




We use a **simple averaging** of **all** the nodes' features:

$$\mathcal{R}(\mathbf{H}) = \sigma\left(\frac{1}{N}\sum_{i=1}^{N}\vec{h}_i\right)$$

## Loss Function




$$\mathcal{L} = \frac{1}{N+M}\left(\sum_{i=1}^{N}\mathbb{E}_{(\mathbf{X},\mathbf{A})}\left[\log \mathcal{D}\left(\vec{h}_i, \vec{s}\right)\right] + \sum_{j=1}^{M}\mathbb{E}_{(\widetilde{\mathbf{X}},\widetilde{\mathbf{A}})}\left[\log\left(1 - \mathcal{D}\left(\vec{\widetilde{h}}_j, \vec{s}\right)\right)\right]\right)$$

最大化互信息

$$\mathcal{D}(\vec{h}_i, \vec{s}) = \sigma\left(\vec{h}_i^T \mathbf{W} \vec{s}\right)$$

# 基于图神经网络的动态网络异常检测算法

郭嘉琰[1]，李荣华[2]，张 岩[1]，王国仁[2]

[1](北京大学 信息科学技术学院,北京 100871)
[2](北京理工大学 计算机学院,北京 100081)
通讯作者: 李荣华, E-mail: lironghuabit@126.com

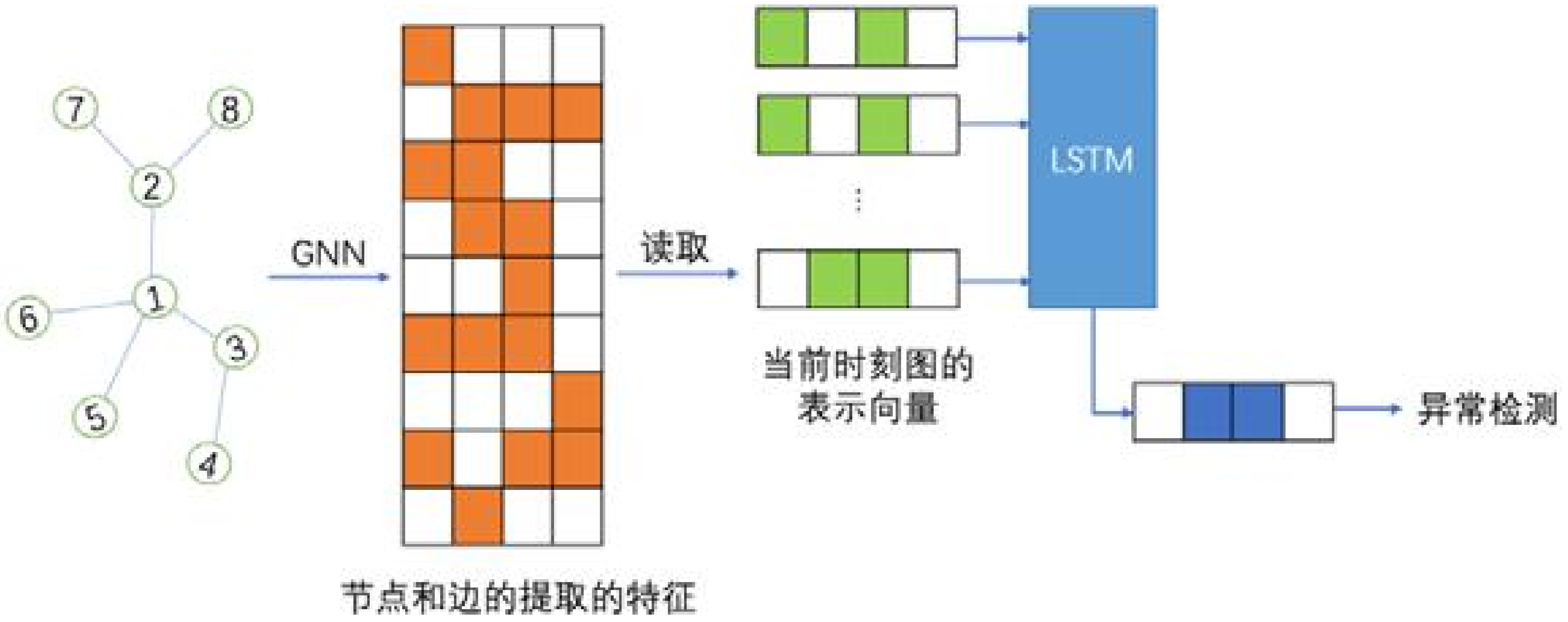
**Unsupervised Learning**

软件学报

# 基于图神经网络的动态网络异常检测算法

使用该算法学得的网络表示向量进行异常检测,得到的结果优于最新的子图异常检测算法SpotLight,并且显著优于传统的网络表示学习算法。

(1) 图的属性特征、结构特征的提取.使用图神经网络来提取某时刻图的属性特征和结构特征;
(2) 图的时间变化特征的提取.使用长短路记忆模型来结合不同时刻图的信息提取图的变化特征;
(3) 动态网络表示学习.使用最大化局部与全局表示互信息的策略来进行图表示向量的学习;
(4) 流数据的异常检测.使用数据流上的异常检测算法给出异常分数.



## Step1: 基于图神经网络的图特征提取

$$Z^{(l+1)} = Act(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}Z^{l}W^{l})$$ $Act(\bullet)$为激活函数

在动态网络数据中,有的情况下,除了节点具有实际意义和属性外,**边也具有实际的意义和属性**,比如通信网络中两个IP 地址之间建立的连接.因此在设计网络结构、属性特征提取器的时候, 要同时考虑边的信息和节点的信息.

将图转换成对应的**线图(line graph)**来获取以边为基本元素的网络:

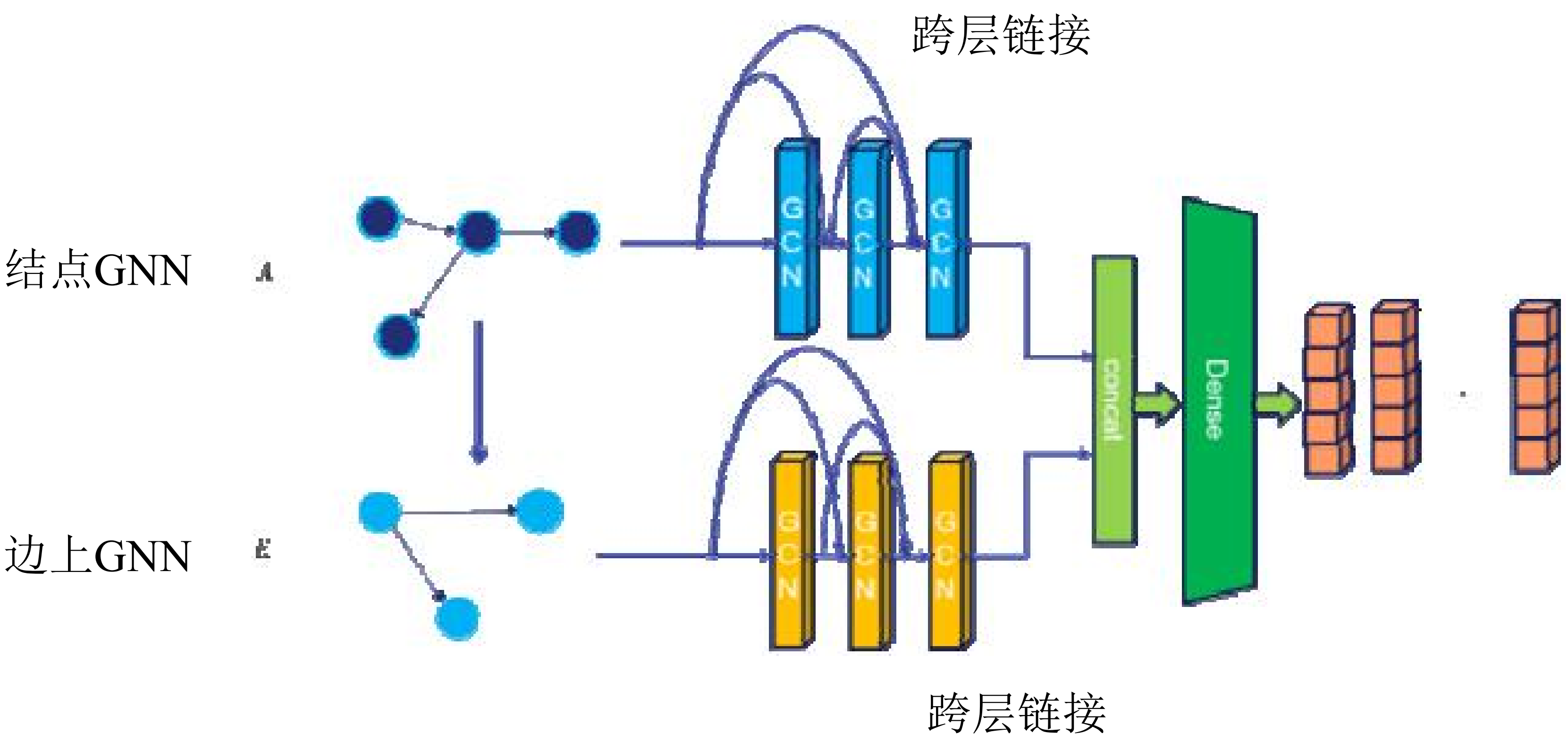
$$E_{ij} = \begin{cases} 1, & e_{i,from} = e_{j,from} \text{ or } e_{i,to} = e_{j,to} \\ 0, & \text{otherwise} \end{cases}$$

$G(V,E)$线图: $L(G)$ $V(L(G)) = E = \{e_1 e_2, \ldots, e_m\}$

$e_i, e_j$是相邻结点等价于两边相连与同一个node

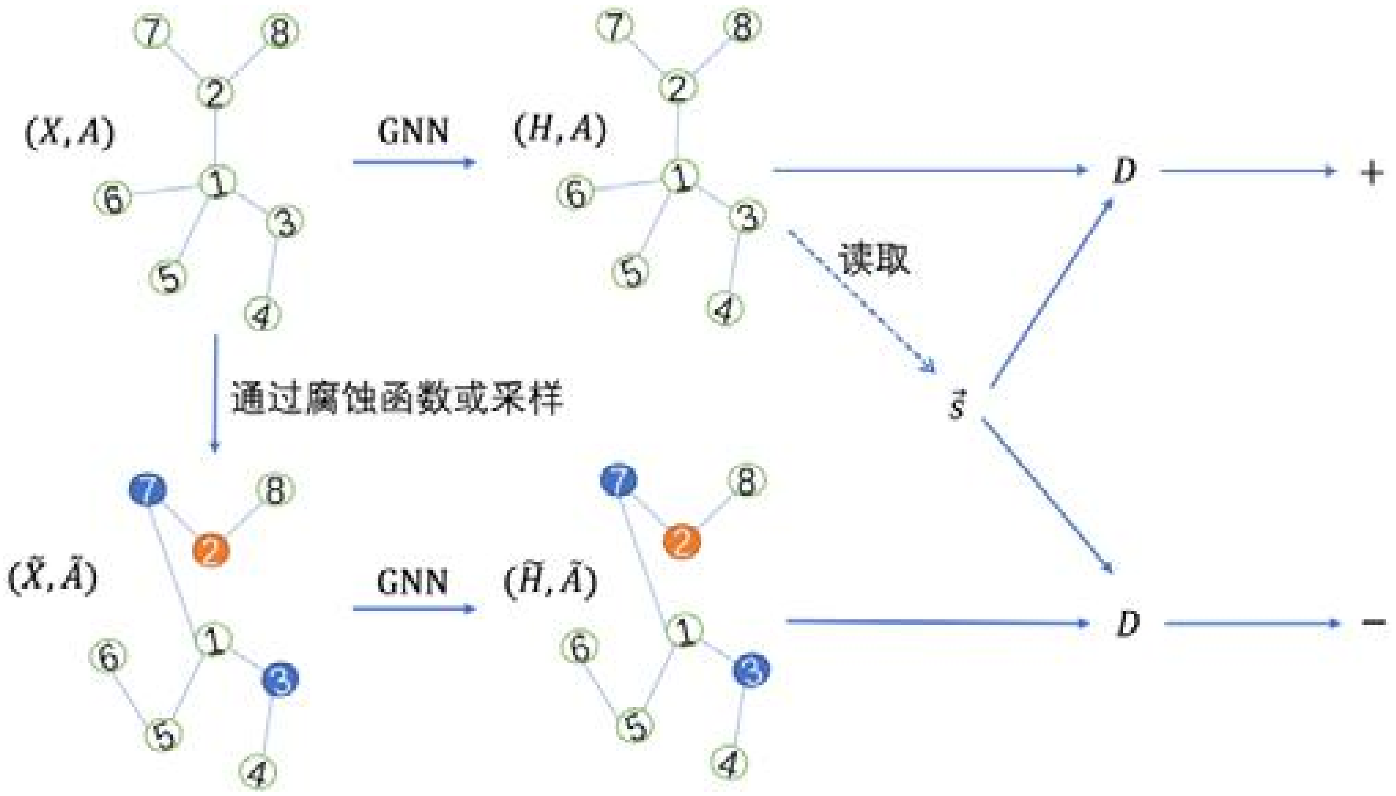
$$Z_E^{(l+1)} = Act(\tilde{D}_E^{-1/2}\tilde{E}\tilde{D}_E^{-1/2}Z_E^{l}W_E^{l})$$

## Step1: 基于图神经网络的图特征提取



# 基于图神经网络的动态网络异常检测算法

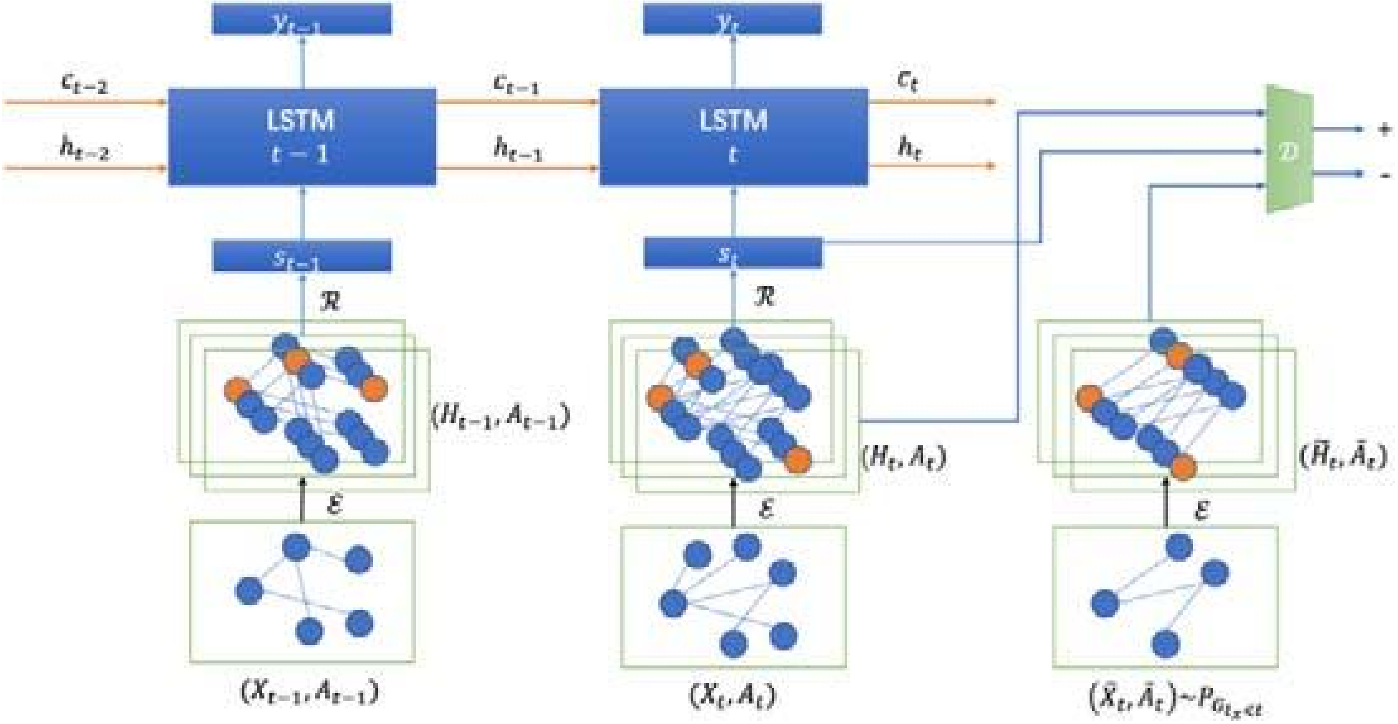
## Step2:基于互信息最大化的网络表示学习




$$L_1 = \frac{1}{N+M}\left(\sum_{i=1}^{N} \mathbb{E}_{(X,A)}[\log \mathcal{D}(\vec{h}_i,\vec{s})] + \sum_{j=1}^{M} \mathbb{E}_{(\tilde{X},\tilde{A})}[\log(1-\mathcal{D}(\tilde{\vec{h}}_j,\vec{s}))]\right)$$

## Step3:基于长短路记忆模型的动态网络表示学习框架

该损失函数使模型最终获得的加入时序信息的表示向量能够尽可能和之前所有的向量的均值相接近。




**假设**模型是**在完全没有异常信息的数据上进行训练的.** 这样,当进行预测任务时,**突然出现的异常子图**的表示向量和其他时刻的子图的表示向量会具有**较大的差距**

$$L_2 = \left\| y_t - \frac{1}{t-1}\sum_{i=1}^{t-1} y_i \right\|_2$$

$$L = \alpha \frac{1}{N+M}\left(\sum_{i=1}^{N} \mathbb{E}_{(X_t,A_t)}[\log \mathcal{D}(\vec{h}_i,\vec{s})] + \sum_{j=1}^{M} \mathbb{E}_{(\tilde{X}_t,\tilde{A}_t)}[\log(1-\mathcal{D}(\vec{\tilde{h}}_j,\vec{s}_t))]\right) + \beta \left\| y_t - \frac{1}{t-1}\sum_{i=1}^{t-1} y_i \right\|_2$$

## Step4: 数据流上的异常检测算法

下游模型接入RRCF、streaming k-means、自编码器

### ·RRCF

**定义 1(RRCT).** 点集 $S$ 上的一个鲁棒随机划分森林由以下步骤产生.

1. 选择一个随机的正比于 $\frac{l_i}{\sum_j l_j}$ 的维度,其中,$l_i=\max_{x\in S}-\min_{x\in S}$;
2. 选择 $X_i\sim Uniform[\min_{x\in S}x_i,\max_{x\in S}x_i]$;
3. 令 $S_1=\{x|x\in S,x_i\leqslant X_i\}$,并且 $S_2=S\backslash S_1$;
4. 在 $S_1$ 和 $S_2$ 上重复步骤 1~步骤 3,直到划分结果为单独的点.

树的编码复杂度可以定义为所有数据点的编码复杂度的和：

$$|M(T')|=\sum_{y\in Z-\{x\}}f(y,Z,T)$$

之后定义数据点的异常分数为删去该异常点后复杂度的减少的期望:

$$score_{anomay}=E_{T(Z)}\|M(T)\|-E_{T(Z-\{x\})}\|M(T(Z-\{x\}))\|$$

## Step4: 数据流上的异常检测算法

·**Streaming *k*-means**

动态更新聚类中心的一种聚类算法,其使用延迟系数(decayfactor)来动态地更新聚类中心:

更新聚类中心.令 $\{x_i\}_{i=1}^{n_0}$ 为已经存在的 $n_0$ 个数据点,此时在时间节点 $t'$ 有 $n'$ 个新的数据 $\{x'_i\}_{i=1}^{n'}$ 到来,新的聚类中心为 $\mathbf{c}$,延迟系数为 $\alpha$,则对应的聚类中心更新为

$$c=\frac{\alpha c_0 n_0+(1-\alpha)\sum_{i=1}^{n'} x'_i}{\alpha n_0+(1-\alpha)n'} \tag{11}$$

之后定义异常分数为数据点到离其最近的聚类中心的距离:

$$score_{anomaly}=\|c_{nearest}-x_i\|_2 \tag{12}$$

# Step4: 数据流上的异常检测算法

## ·Encoder-Decoder-Encoder

因此可以使用这种架构的异常检测器来帮助完成异常检测.首先,Encoder-Decoder-Encoder 框架中的第 1 个 Encoder 是将实体编码为分布式向量的函数(神经网络).当信息被编码为分布式向量后,我们利用分布式向量训练一个 Decoder,使该分布式向量能够很好地吸收原信息中最有用的信息.此时需要重建误差(reconstrcuterror)作为模型的损失:

$$\mathcal{L}_{reconstruct}=\|Decoder(Encoder(X))-X\|_2 \tag{13}$$

当向量被 Decoder 解码之后,将解码后的向量送入一个新的 Encoder 中.这个 Encoder 需要和之前的 Encoder 的结构保持一致.在设计损失函数时,加入该 Encoder 和之前 Encoder 得出的表示向量之间的距离来让第 2 个 Encoder 尽可能去拟合第一个 Encoder 得到的结果,这就引入了拟合误差:

$$\mathcal{L}_{fit}=\|Encoder(Decoder(Encoder(X)))-Encoder(X)\|_2 \tag{14}$$

这样,当新的数据到来时,如果是和之前的数据分布相同的数据,第 1 个 Encoder 和第 2 个 Encoder 结果之间的差距会尽可能小;而当异常的数据到来时,因为第 2 个 Encoder 从来没有见过异常的数据,就会使两个 Encoder 之间的误差变大.因此,可以直接使用两个 Encoder 之间的误差作为异常分数:

$$score_{anomaly}=\|Encoder(Decoder(Encoder(X)))-Encoder(X)\|_2 \tag{15}$$

这种做法不需要存储聚类中心或查找距离新数据最近的聚类中心,只需保存模型即可.模型的两个输出之间的差可以直接作为最终的异常分数被使用.

# AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN

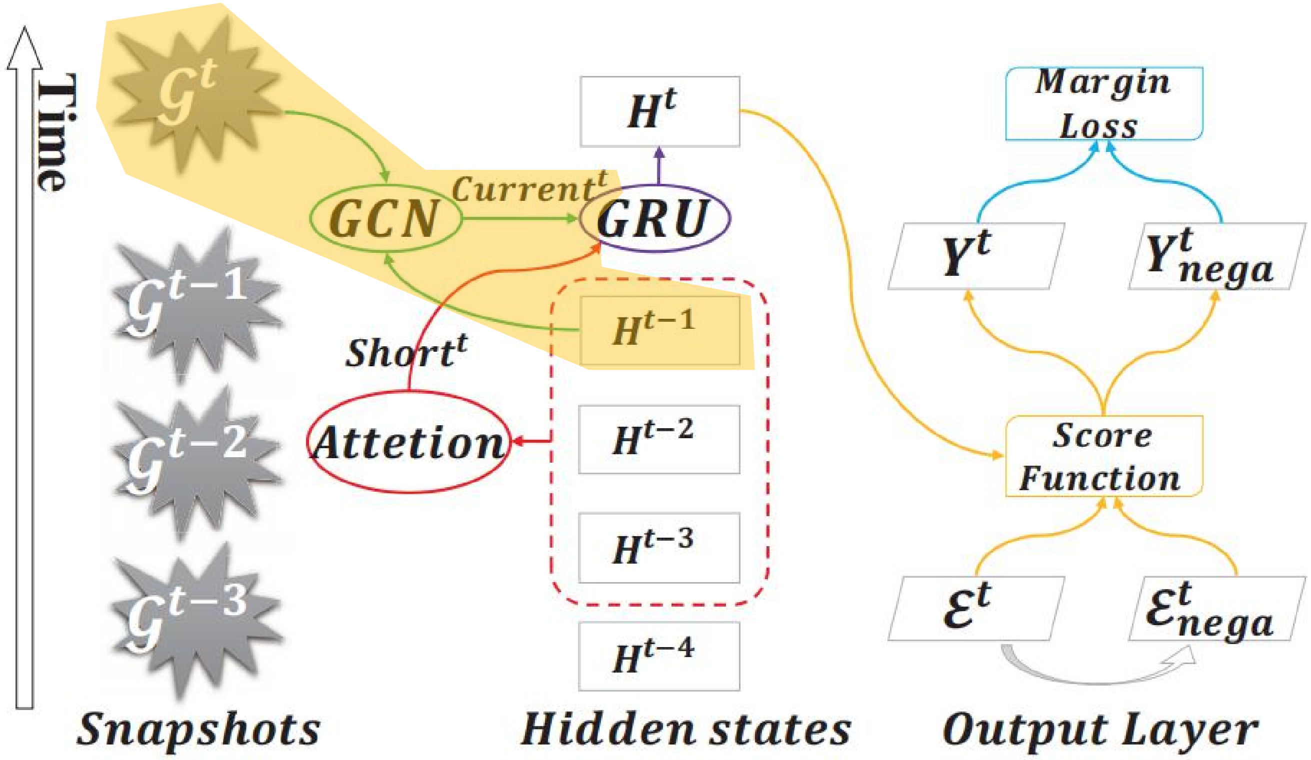
AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN

**Li Zheng**[1,2], **Zhenpeng Li**[3], **Jian Li**[3], **Zhao Li**[3*] and **Jun Gao**[1,2*]
[1]The Key Laboratory of High Confidence Software Technologies, Ministry of Education, China
[2]School of EECS, Peking University, China
[3]Alibaba Group, China
{greezheng, gaojun}@pku.edu.cn, {zhen.lzp,zeshan.lj,lizhao.lz}@alibaba-inc.com

**IJCAI-19**

动态属性图的异常检测

# AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN




**GCN for content and structural features.** At timestamp $t$, we receive the snapshot $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$ with its adjacency matrix $\mathbf{A}^t$ and the output hidden state matrix $\mathbf{H}^{t-1} \in \mathbb{R}^{n\times d}$ of the framework at timestamp $t-1$. First, we propagate the hidden state matrix with GCN,

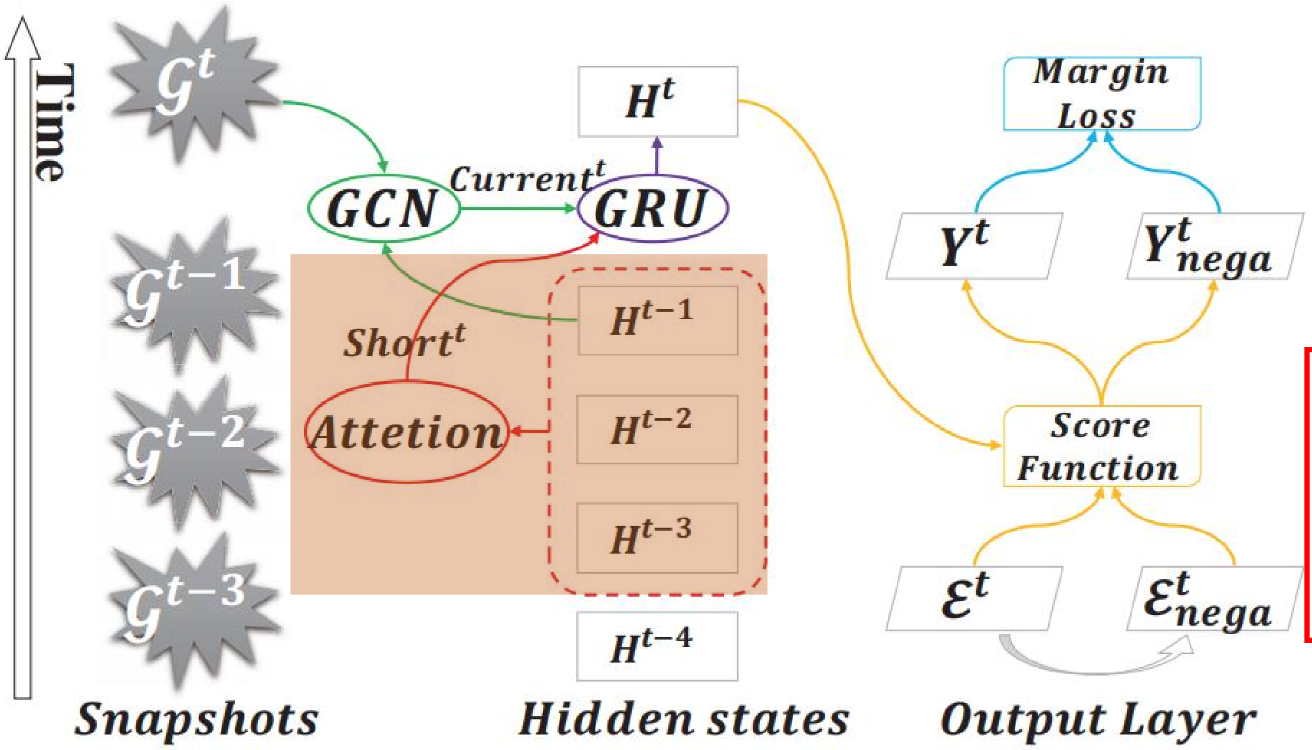
$$\mathbf{Current}^t = \mathrm{GCN}_L(\mathbf{H}^{t-1}), \quad (1)$$

where $\mathbf{Current}^t$ represents current state of nodes combining the current input with the long-term hidden state, and $\mathrm{GCN}_L$ denotes an $L$-layered GCN which is proposed in [Kipf and

$$\begin{aligned}
\mathbf{Z}^{(0)} &= \mathbf{H}^{t-1}, \\
\mathbf{Z}^{(l)} &= ReLU(\hat{\mathbf{A}}^t \mathbf{Z}^{(l-1)} \mathbf{W}^{(l-1)}), \\
\mathbf{Current}^t &= ReLU(\hat{\mathbf{A}}^t \mathbf{Z}^{(L-1)} \mathbf{W}^{(L-1)}),
\end{aligned}$$

$$\hat{\mathbf{A}}^t = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}}^t \tilde{\mathbf{D}}^{-\frac{1}{2}}$$

# AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN




**GRU with attention to combine short-term and long-term states.** To catch the short-term pattern of nodes, we apply the contextual attention-based model which is inspired by [Liu *et al.*, 2017] and proposed by [Cui *et al.*, 2017]. In our framework, we construct short state of local window as follow:

$$\mathbf{C}^t_{h,i} = [\mathbf{h}^{t-\omega}_i; \dots; \mathbf{h}^{t-1}_i] \qquad \mathbf{C}^t_{h,i} \in \mathbb{R}^{\omega \times d} \qquad (5)$$

$$\mathbf{e}^t_{h,i} = \mathbf{r}^T \tanh(\mathbf{Q}_h (\mathbf{C}^t_{h,i})^T) \qquad \mathbf{e}^t_{h,i} \in \mathbb{R}^{\omega} \qquad (6)$$

$$\mathbf{a}^t_{h,i} = softmax(\mathbf{e}^t_{h,i}) \qquad \mathbf{a}^t_{h,i} \in \mathbb{R}^{\omega} \qquad (7)$$

$$\mathbf{short}^t_i = (\mathbf{a}_{h,i} \mathbf{C}^t_{h,i})^T \qquad \mathbf{short}^t_i \in \mathbb{R}^{d} \qquad (8)$$

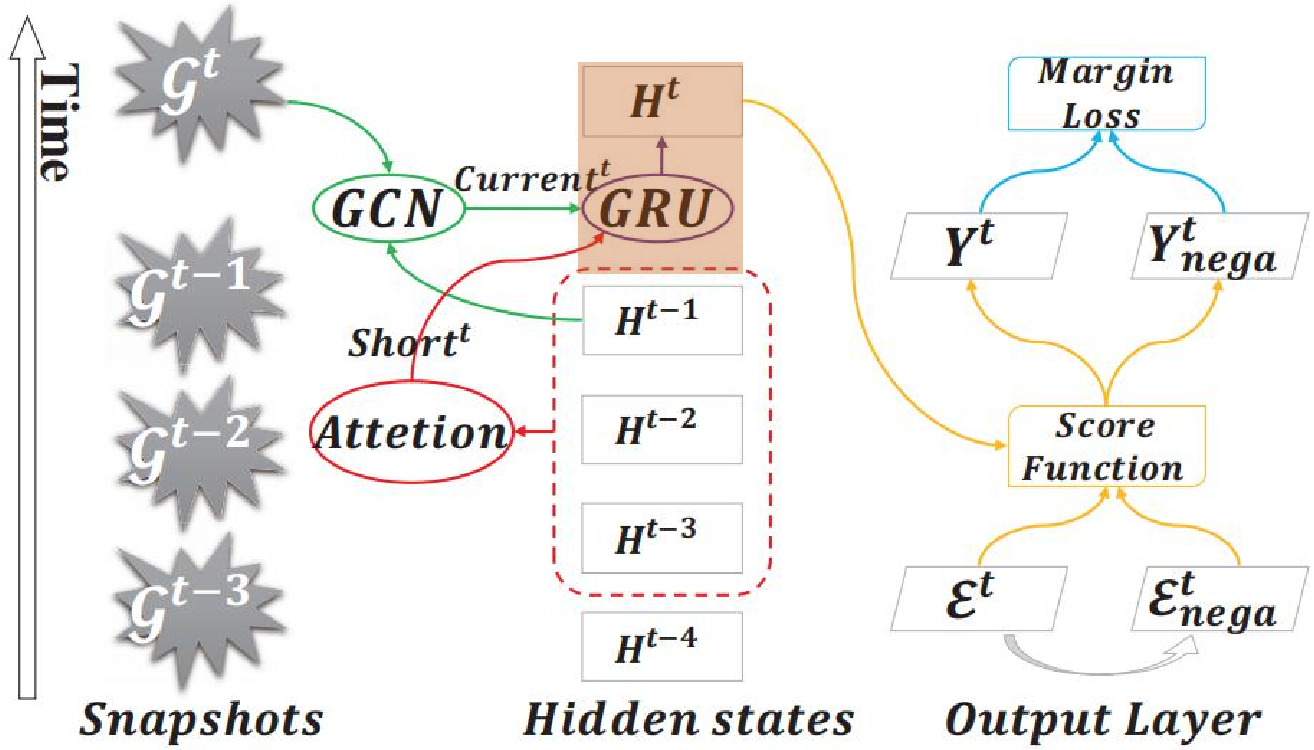
**Brief**

$$\mathbf{short}^t_i = \mathrm{CAB}(\mathbf{h}^{t-\omega}_i; \dots; \mathbf{h}^{t-1}_i)$$

$$\mathbf{Short}^t = \mathrm{CAB}(\mathbf{H}^{t-\omega}; \dots; \mathbf{H}^{t-1})$$

# AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN

## GRU Framework




$$\mathbf{H}^t = \mathrm{GRU}(\mathbf{Current}^t, \mathbf{Short}^t)$$

$$\mathbf{P}^t = \sigma(\mathbf{U}_P\mathbf{Current}^t + \mathbf{W}_P\mathbf{Short}^t + \mathbf{b}_P)$$
$$\mathbf{R}^t = \sigma(\mathbf{U}_R\mathbf{Current}^t + \mathbf{W}_R\mathbf{Short}^t + \mathbf{b}_R)$$
$$\tilde{\mathbf{H}}^t = \tanh(\mathbf{U}_c\mathbf{Current}^t + \mathbf{W}_c(\mathbf{R}^t \odot \mathbf{Short}^t))$$
$$\mathbf{H}^t = (1 - \mathbf{P}^t) \odot \mathbf{Short}^t + \mathbf{P}^t \odot \tilde{\mathbf{H}}^t$$

## Anomalous score computation for edges

**For each edge (*i, j, w*), anomalous scores:**

$$f(i, j, w) = w \cdot \sigma(\beta \cdot (||\mathbf{a} \odot \mathbf{h}_i + \mathbf{b} \odot \mathbf{h}_j||_2^2 - \mu)) \quad (16)$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ are the hidden state of the $i$-th and $j$-th node respectively, and $\sigma(x) = \frac{1}{1+e^x}$ is the sigmoid function. $\mathbf{a}$ and $\mathbf{b}$ are parameters to optimize in the output layer. $\beta$ and $\mu$ are the hyper-parameters in the score function. Note that the single layer network used in this paper can be replaced by other sophisticated networks.

# AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN

## Negative Sampling

ple as an anomalous edge. Inspired by the method proposed in [Wang *et al.*, 2014], we define a Bernoulli distribution with parameter $\frac{d_i}{d_i+d_j}$ for sampling: given a normal edge $(i, j)$, we replace $i$ with probability $\frac{d_i}{d_i+d_j}$ and replace $j$ with probability $\frac{d_j}{d_i+d_j}$, where $d_i$ and $d_j$ denote the degree of the $i$-th node and the $j$-th node respectively.

## Loss Function

$$\mathcal{L} = \mathcal{L}^t + \lambda \mathcal{L}_{reg}$$

$$\mathcal{L}^t = \min \sum_{(i,j,w)\in\mathcal{E}^t} \sum_{(i',j',w)\notin\mathcal{E}^t} \max\{0, \gamma + f(i,j,w) - f(i',j',w)\}$$

$$\begin{aligned}\mathcal{L}_{reg} = \sum(&||\mathbf{W}_1||_2^2 + ||\mathbf{W}_2||_2^2 + ||\mathbf{Q}_h||_2^2 + ||\mathbf{r}||_2^2 \\ &+||\mathbf{U}_z||_2^2 + ||\mathbf{W}_z||_2^2 + ||\mathbf{b}_z||_2^2 + ||\mathbf{U}_r||_2^2 + ||\mathbf{W}_r||_2^2 \\ &+||\mathbf{b}_r||_2^2 + ||\mathbf{U}_c||_2^2 + ||\mathbf{W}_c||_2^2 + ||\mathbf{a}||_2^2 + ||\mathbf{b}||_2^2) \quad (19)\end{aligned}$$

# Spotting Terrorists by Learning Behavior-aware Heterogeneous Network Embedding

Spotting Terrorists by Learning Behavior-aware Heterogeneous Network Embedding

Pei-Chi Wang
Department of Statistics
National Cheng Kung University
Tainan, Taiwan
fcps974125@gmail.com

Cheng-Te Li
Institute of Data Science
National Cheng Kung University
Tainan, Taiwan
chengte@mail.ncku.edu.tw

**CIKM-19**

# Spotting Terrorists by Learning Behavior-aware Heterogeneous Network Embedding

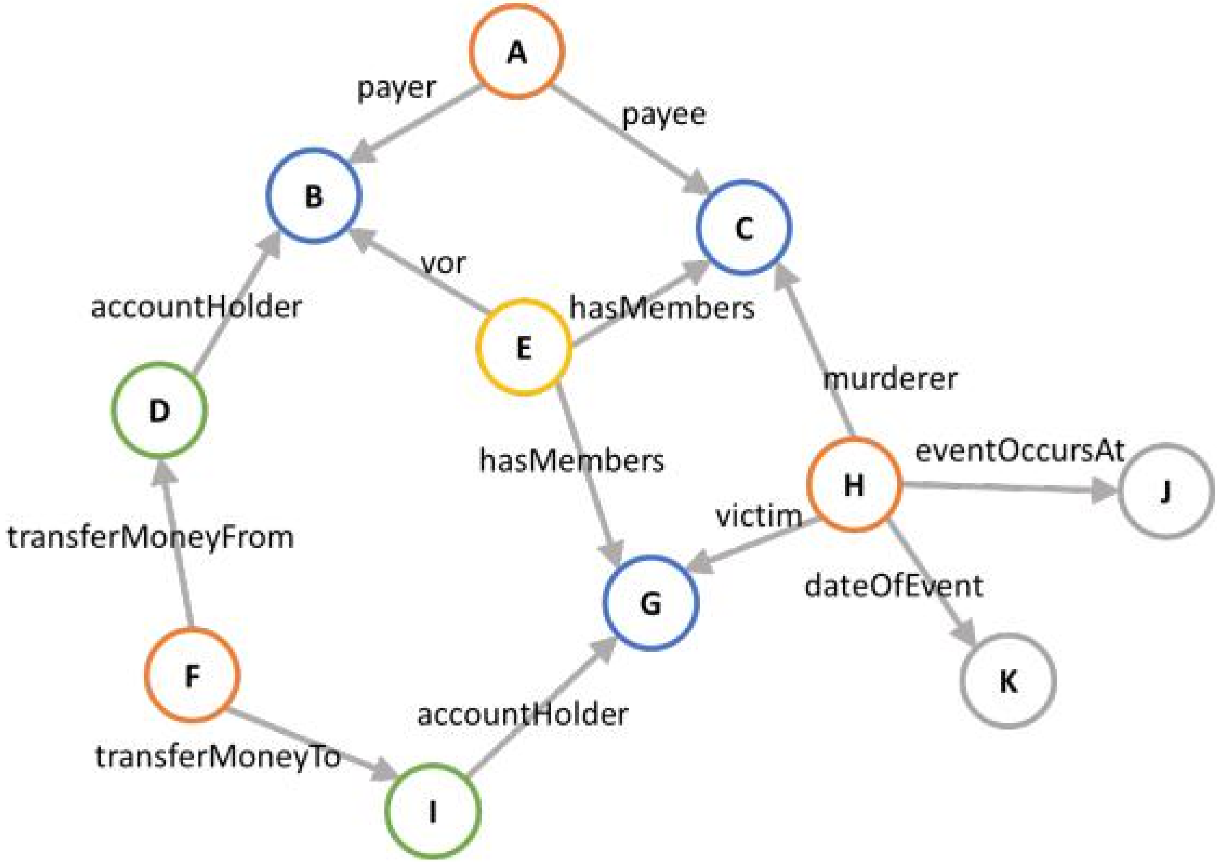



**Figure 1: A toy example of a heterogeneous criminal network. Different colors on nodes indicate different types of nodes. Edges are associated with different typed labels.**

Figure 1 is a toy example of heterogeneous criminal network. An exampled triple $(A, payee, C)$ is used to represent the "*payee*" relation from entity $A$ to $C$, i.e., $A \xrightarrow{payee} C$. Another exampled triple $(C, \langle murderer^{-1}, victim \rangle, G)$ is used to represent the indirect relationships from entity $C$ to $G$, i.e., $C \xleftarrow{murderer} \xrightarrow{victim} G$.

# Spotting Terrorists by Learning Behavior-aware Heterogeneous Network Embedding

## Learn Behavior-aware Entity Embedding

Given a heterogeneous criminal network $\mathcal{K} = (\mathcal{E}, \varphi, \mathcal{R}, \omega, \mathcal{T})$, along with all triples of relation paths $\mathcal{P} = (h, p, t) \in \mathcal{K}$, we aim at learning the embedding matrices $\mathbf{E} \in \mathbb{R}^{n_e \times d}$ and $\mathbf{P} \in \mathbb{R}^{n_p \times d}$ for all entities and all relation paths, respectively, where $n_e$ is the numbers of entities and $n_p$ is the number of relation paths, and $d$ is the dimensionality of embedding vectors. Let $\mathbf{x}_h$, $\mathbf{x}_p$, and $\mathbf{x}_t$ be the index one-hot vectors for a head-relation-tail triple $(h, p, t)$. Hence, the embedding vectors for head entity $h$, relation path $p$, and tail entity $t$ are $\mathbf{h} = \mathbf{x}_h^\top \mathbf{E}$, $\mathbf{p} = \mathbf{x}_p^\top \mathbf{P}$, and $\mathbf{t} = \mathbf{x}_t^\top \mathbf{E}$. To effectively learn

## Scoring Function

$$\bar{\mathbf{t}} = \mathbf{h} \circ \mathbf{p}.$$

***Hadamard product***

and the learned tail embedding $\mathbf{t}$. The scoring function is:

$$s(h, p, t) = tanh(\bar{\mathbf{t}}^\top \mathbf{t}),$$

where $tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$.

# Spotting Terrorists by Learning Behavior-aware Heterogeneous Network Embedding

## Behavior Penalty

$$\pi_h = deg(h)$$ *Degree based*

or

$$\pi_h = -\sum_{i \in \Psi(h)} P(i) \log P(i)$$ *Entropy based*

## Loss Function

$$\mathcal{L} = \sum_{p \in \rho} \sum_{(h,p,t) \in \mathcal{P}} \sum_{(h',p,t') \in \mathcal{P}'} \left[ -\frac{s(h,p,t)}{\pi_h} + \underbrace{\frac{s(h',p,t')}{\pi_{h'}}}_{\text{Negative Sampling}} \right] + \lambda \sum \underbrace{\|\theta\|_2^2}_{\text{L2 regularization}},$$

**Instead of** random negative **sampling over all entities**, we perform random **sampling over entities of each type**, and ensure every entity type will be sampled.

# Heterogeneous Graph Neural Networks for Malicious Account Detection

Ziqi Liu
Ant Financial Services Group
Hangzhou, China
ziqiliu@antfin.com

Chaochao Chen
Ant Financial Services Group
Hangzhou, China
chaochao.ccc@antfin.com

Xinxing Yang
Ant Financial Services Group
Beijing, China
xinxing.yangxx@antfin.com

Jun Zhou
Ant Financial Services Group
Beijing, China
jun.zhoujun@antfin.com

Xiaolong Li
Ant Financial Services Group
Seattle, USA
xl.li@antfin.com

Le Song
Georgia Institute of Technology
Ant Financial Services Group
GA, USA
lsong@cc.gatech.edu

**CIKM-18**

蚂蚁金服异常检测

# Heterogeneous Graph Neural Networks for Malicious Account Detection

## Heterogeneous Graph Construction

a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

For our convenience, we can further extract $|\mathcal{D}|$ subgraphs $\{\mathcal{G}^{(d)} = (\mathcal{V}, \mathcal{E}^{(d)})\}$ each of which preserves all the vertices of $\mathcal{G}$, but ignores the edges containing devices that do not belong to type $d$. This leads to $|\mathcal{D}|$ adjacency matrices $\{A^{(d)}\}$. Note that the heterogeneous graph representation $\{\mathcal{G}^{(d)}\}$ lies in the same storage complexity compared with original $\mathcal{G}$ because we only need to store the sparse edges.

**异构图：users, items**

**Items的类型有6种**

**账户行为 在将时间窗(0,T)等分成p段，统计账户在每段的行为次数作为特征（p维）**

**设备类型one-hot编码：(|D|维)**

Along with these graphs, we can further observe the activities of each account. Assuming a $N$ by $p + |\mathcal{D}|$ matrix $X \in \mathbb{R}^{N, p+|\mathcal{D}|}$, with each row $x_i$ denotes activities of vertex $i$ if $i$ is an account. In practice, the activities of account $i$ over a time period $[0, T)$ can be discretized into $p$ time slots, where the value of each time slot denotes the count of the activities in this time slot. For vertices correspond to devices, we just encode $x_i$ as one hot vector using the last $|\mathcal{D}|$ coordinates.

# Heterogeneous Graph Neural Networks for Malicious Account Detection

## Feed Forward

$$H^{(0)} \leftarrow \mathbf{0}$$

for $t = 1, \ldots, T$ T是网络层数

$$H^{(t)} \leftarrow \sigma\left(X \cdot W + \frac{1}{|\mathcal{D}|} \sum_{d=1}^{|\mathcal{D}|} A^{(d)} \cdot H^{(t-1)} \cdot V_d\right)$$

## Optimization

取值+1, -1

$$\min_{W, \{V_d\}, u} \mathcal{L}(W, \{V_d\}, u) = -\sum_{i}^{N_o} \log \sigma(y_i \cdot (u^\top h_i))$$

where $\sigma$ denotes logistic function $\sigma(x) = \frac{1}{1+\exp -x}$, $u \in \mathbb{R}^k$, and the loss $\mathcal{L}$ sums over partially observed $N_o$ accounts with known labels. Our algorithm works interatively in an Expectation Maximization style. In e-step, we compute the embeddings based on current parameters $W, \{V_d\}$ as in Eq (6). In m-step, we optimize those parameters in Eq (7) while fixing embedings.

**E-M风格学习策略：**

**1.e-step：基于已有参数，计算embedding；**

**2.m-step:固定eembedding，优化参数。**

# Heterogeneous Graph Neural Networks for Malicious Account Detection

## With Attention

$$H^{(t)} \leftarrow \sigma\left(X \cdot W + \frac{1}{|\mathcal{D}|} \sum_{d=1}^{|\mathcal{D}|} A^{(d)} \cdot H^{(t-1)} \cdot V_d\right)$$

$$\downarrow$$

$$H^{(t)} \leftarrow \sigma\left(X \cdot W + \sum_{d \in \mathcal{D}} \mathrm{softmax}(\alpha_d) \cdot \quad A^{(d)} \cdot H^{(t-1)} \cdot V_d\right)$$

# Deep Structure Learning for Fraud Detection

Haibo Wang*‡§, Chuan Zhou†, Jia Wu¶, Weizhen Dang*‡§, Xingquan Zhu‖, Jilong Wang‡§
*Department of Computer Science and Technology, Tsinghua University
†Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
‡Institute for Network Sciences and Cyberspace, Tsinghua University
§Tsinghua National Laboratory for Information Science and Technology
¶Department of Computing Faculty of Science and Engineering, Macquarie University, Sydney, Australia
‖ Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, USA
Email: {wang-hb15, dangwz16}@mails.tsinghua.edu.cn, zhouchuan@iie.ac.cn, jia.wu@mq.edu.au, xzhu3@fau.edu, wjl@cernet.edu.cn

**ICDM-18**

# Deep Structure Learning for Fraud Detection

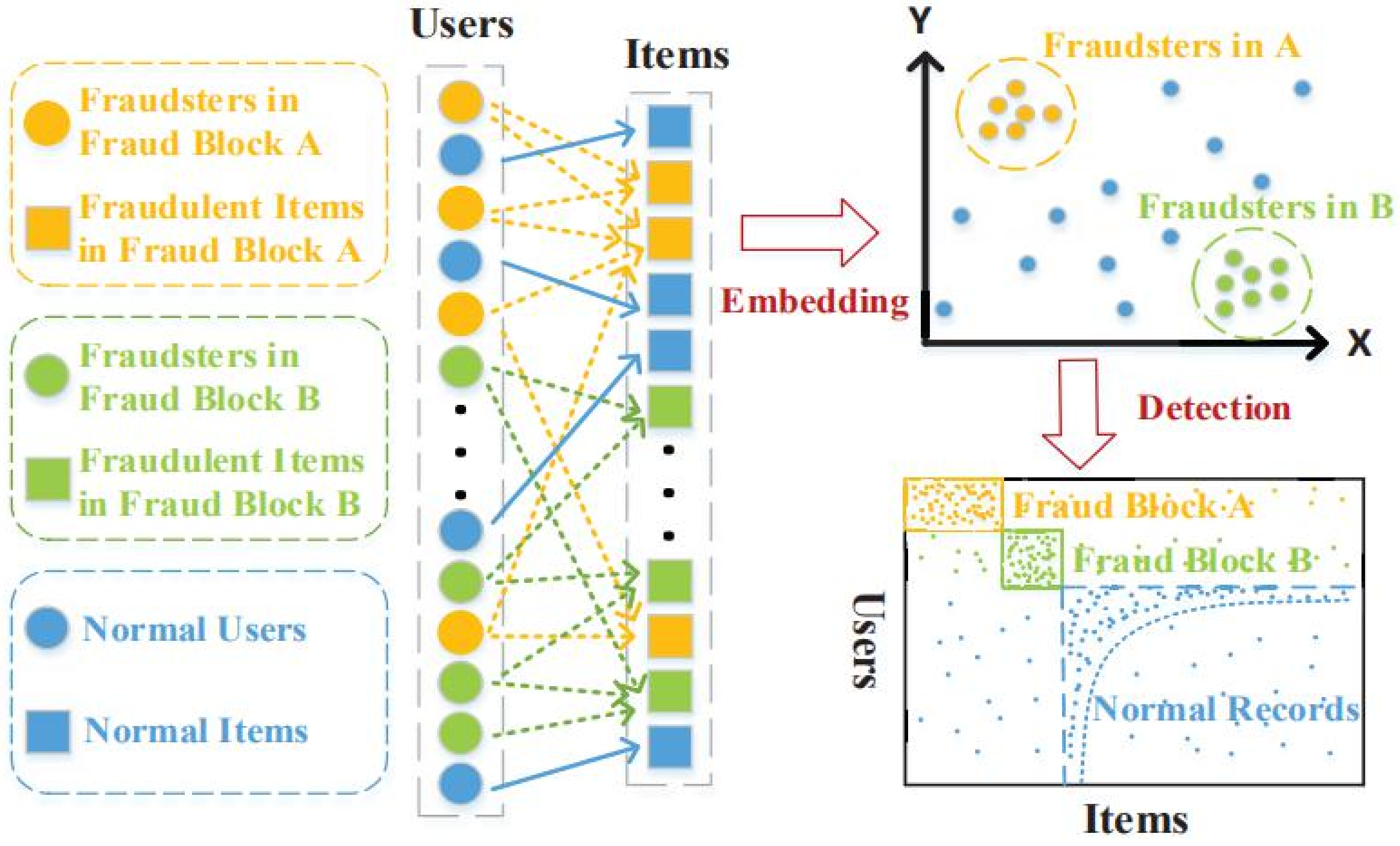



Figure 1. Deep Structure Learning for Fraud Detection. Blue circles and squares represent normal users and items. Yellow/Green ones represent fraudsters and corresponding fraudulent items in one/another fraud block.

# Deep Structure Learning for Fraud Detection

***Definition 1:*** (Interaction Information Graph) An interaction information graph is a special form of a bipartite graph, which is defined as $G = (X, Y, E)$, where $X = \{x_1, ..., x_m\}$ represents $m$ user nodes, $Y = \{y_1, ..., y_n\}$ represents $n$ item nodes and $E = \{e_{ij}\}_{i=1,...,m}^{j=1,...,n}$ represents directed edges from $X$ to $Y$. If there exits an edge from $x_i$ to $y_j$, $e_{ij} = 1$. Otherwise, $e_{ij} = 0$. Note that $X$ and $Y$ are two disjoint sets.

构建结点之间的相似度指标衡量

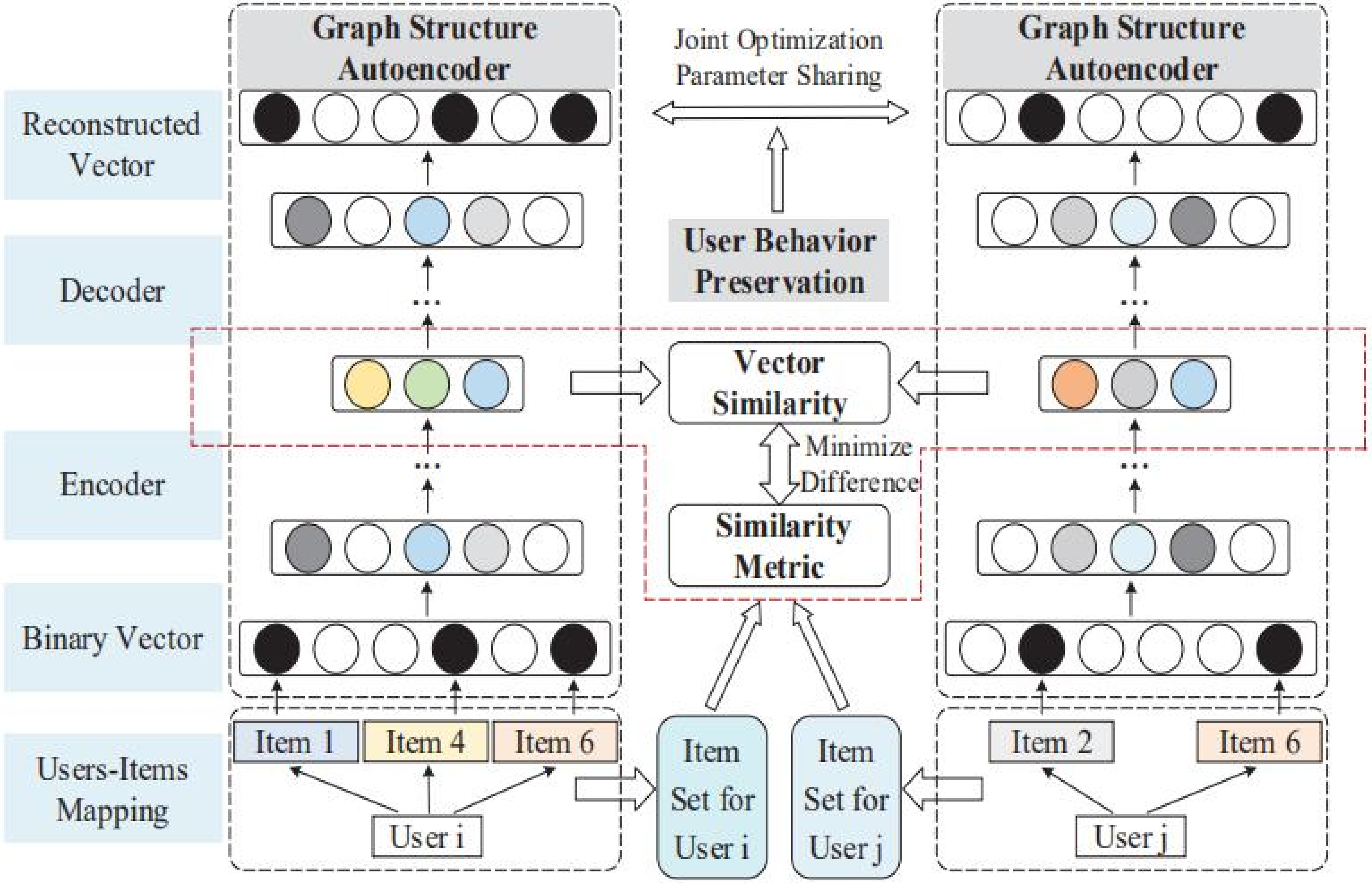
***Definition 2:*** (Similarity Metric) Given two different user nodes $x_i$ and $x_j$ in the interaction information graph, the similarity metric between them can be defined as $sim_{ij} = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$, where $N_i := \{y_j \in Y : e_{ij} = 1\}$ represents the item node set associated with the user node $x_i$.

$$sim_{ij} = \begin{cases} \frac{|N_i \cap N_j|+1}{|N_i \cup N_j|+n} & |N_i \cap N_j| = \varnothing, \\ \frac{|N_i \cap N_j|+(n-1)}{|N_i \cup N_j|+n} & |N_i \cap N_j| = |N_i \cup N_j|), \\ \frac{|N_i \cap N_j|}{|N_i \cup N_j|} & otherwise. \end{cases}$$

Suspicious user nodes inevitably **associate with more of the same item nodes** so that the **similarity between them is relatively higher**, while the behaviors of normal user nodes are independent, which leads to low similarity in general.

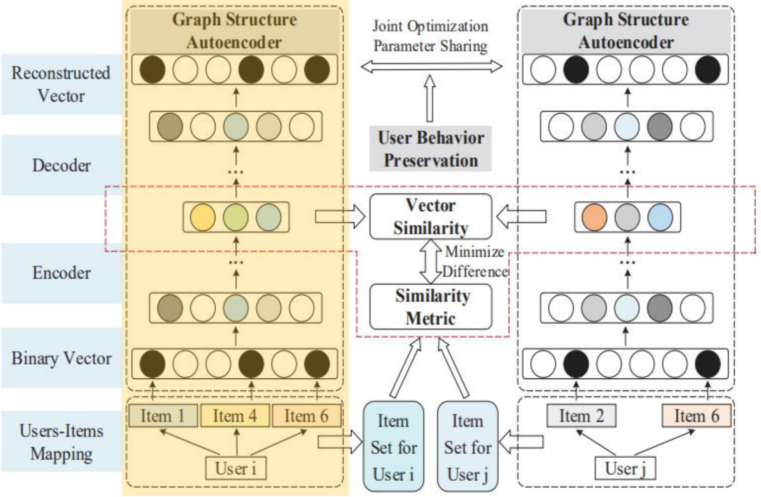
# Deep Structure Learning for Fraud Detection

## DeepFD model



# Deep Structure Learning for Fraud Detection

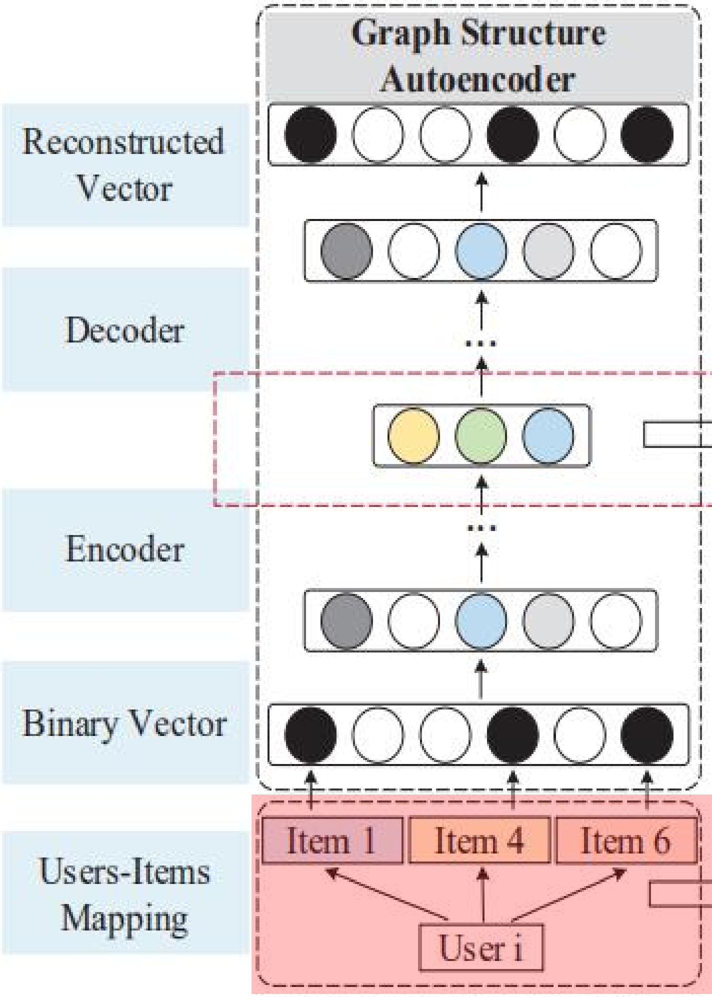
## DeepFD model




$$g_i^{(1)} = \sigma(W^{(1)} s_i + b^{(1)})$$
$$g_i^{(l)} = \sigma(W^{(l)} g_i^{(l-1)} + b^{(l)}), l = 2, ..., K$$

Reconstructed loss function:

$$\mathcal{L}_{tmp} = \sum_{i=1}^{m} ||\hat{s_i} - s_i||_2^2$$




Treats all elements of the input vector $s_i$ equally and the number of zero elements in $s_i$ is **far more than** that of non-zero elements, the auto-encoder is more likely to reconstruct zero elements.

setting larger weights to non-zero elements:

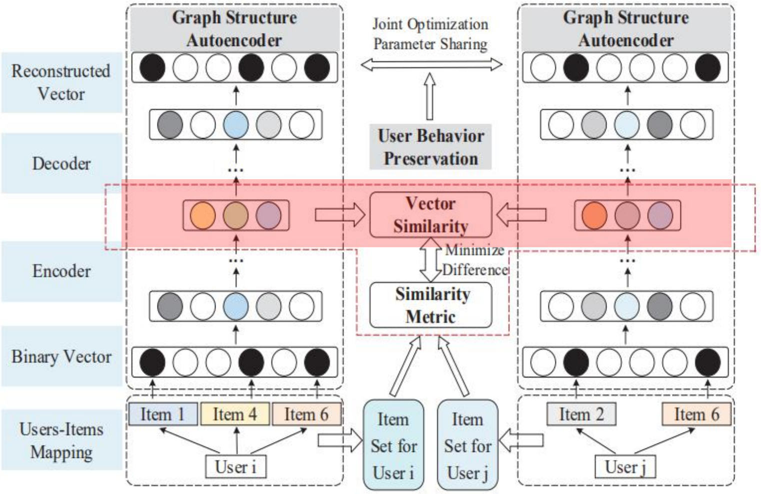
$$\mathcal{L}_{recon} = \sum_{i=1}^{m} ||(\hat{s_i} - s_i) \odot h_i||_2^2 = ||(\hat{S} - S) \odot H||_2^2$$

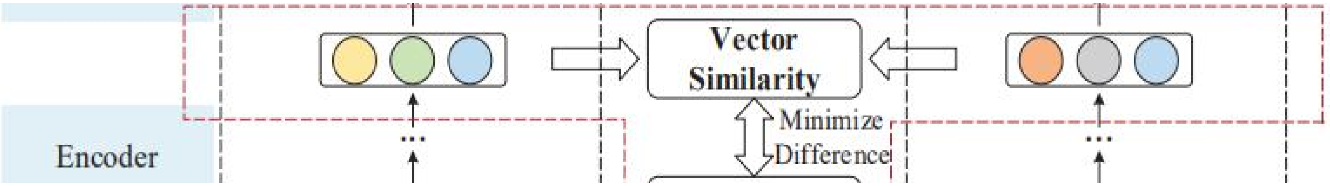
where $\odot$ is the Hadamard product, $\hat{S} = \{\hat{s_1}, \hat{s_2}, ..., \hat{s_m}\}$, $H = \{h_1, h_2, ..., h_m\}$ and $h_i$ is the weight vector for the input vector $s_i$. For $h_i = \{h_{ij}\}_{j=1}^{n}$, if $s_{ij} = 0$, $h_{ij} = 1$; otherwise, $h_{ij} = \beta > 1$.

根据Graph做向量

# Deep Structure Learning for Fraud Detection

## DeepFD model




实际做的时候加入了negative sampling

For user node $x_i$ and user node $x_j$, the distance measure of their vector representations is defined as follows:

$$dis_{ij} = ||(g_i^{(K)} - g_j^{(K)})||_2^2 \tag{4}$$

$$\widehat{sim}_{ij} = \exp\left(-\lambda \cdot dis_{ij}\right)$$

where $\lambda \geq 0$. When the distance of the two user nodes is close to 0, the value of $\widehat{sim}_{ij}$ is close to 1, which means that their vector representations are very similar. While when the distance is large enough, the value of $\widehat{sim}_{ij}$ is close to 0, which means that their vector representations are quite different. In Section II, we have defined an empirical

$$\mathcal{L}'_{sim} = \sum_{i,j=1}^{m} ||\widehat{sim}_{ij} - sim_{ij}||_2^2$$

$$\mathcal{L}_{reg} = \frac{1}{2}\sum_{l=1}^{K}(||W^{(l)}||_2^2 + ||\hat{W}^{(l)}||_2^2 + ||b^{(l)}||_2^2 + ||\hat{b}^{(l)}||_2^2)$$

$$\mathcal{L} = \mathcal{L}_{recon} + \alpha\mathcal{L}_{sim} + \gamma\mathcal{L}_{reg}$$

# Deep Structure Learning for Fraud Detection

## 对于嵌入的向量，采用DBSCAN算法做clustering

## Baseline：

**M-Zoom**

**D-Cube**

**HoloScope**

**FRAUDAR**

**DeepWalk+DBSACN**

**node2vec+DBSACN**

**LINE+DBSACN**


(a) WalkFD1 (b) WalkFD2 (c) LineFD (d) DeepFD

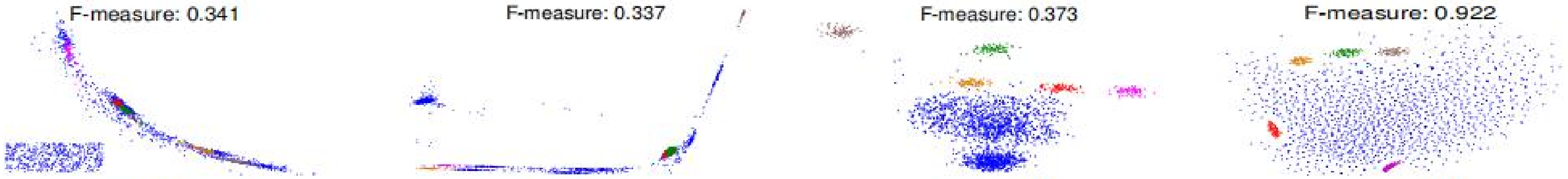
Figure 4. F-measure comparison for different fraud detection methods and visualization for embedding results (**2 fraud blocks injected**). Blue color represents normal users, and other colors represent different fraud blocks.


(a) WalkFD1 (b) WalkFD2 (c) LineFD (d) DeepFD

Figure 5. F-measure comparison for different fraud detection methods and visualization for embedding results (**3 fraud blocks injected**). Blue color represents normal users, and other colors represent different fraud blocks.


(a) WalkFD1 (b) WalkFD2 (c) LineFD (d) DeepFD

Figure 6. F-measure comparison for different fraud detection methods and visualization for embedding results (**4 fraud blocks injected**). Blue color represents normal users, and other colors represent different fraud blocks.




(a) WalkFD1 (b) WalkFD2 (c) LineFD (d) DeepFD

Figure 7. F-measure comparison for different fraud detection methods and visualization for embedding results (**5 fraud blocks injected**). Blue color represents normal users, and other colors represent different fraud blocks.

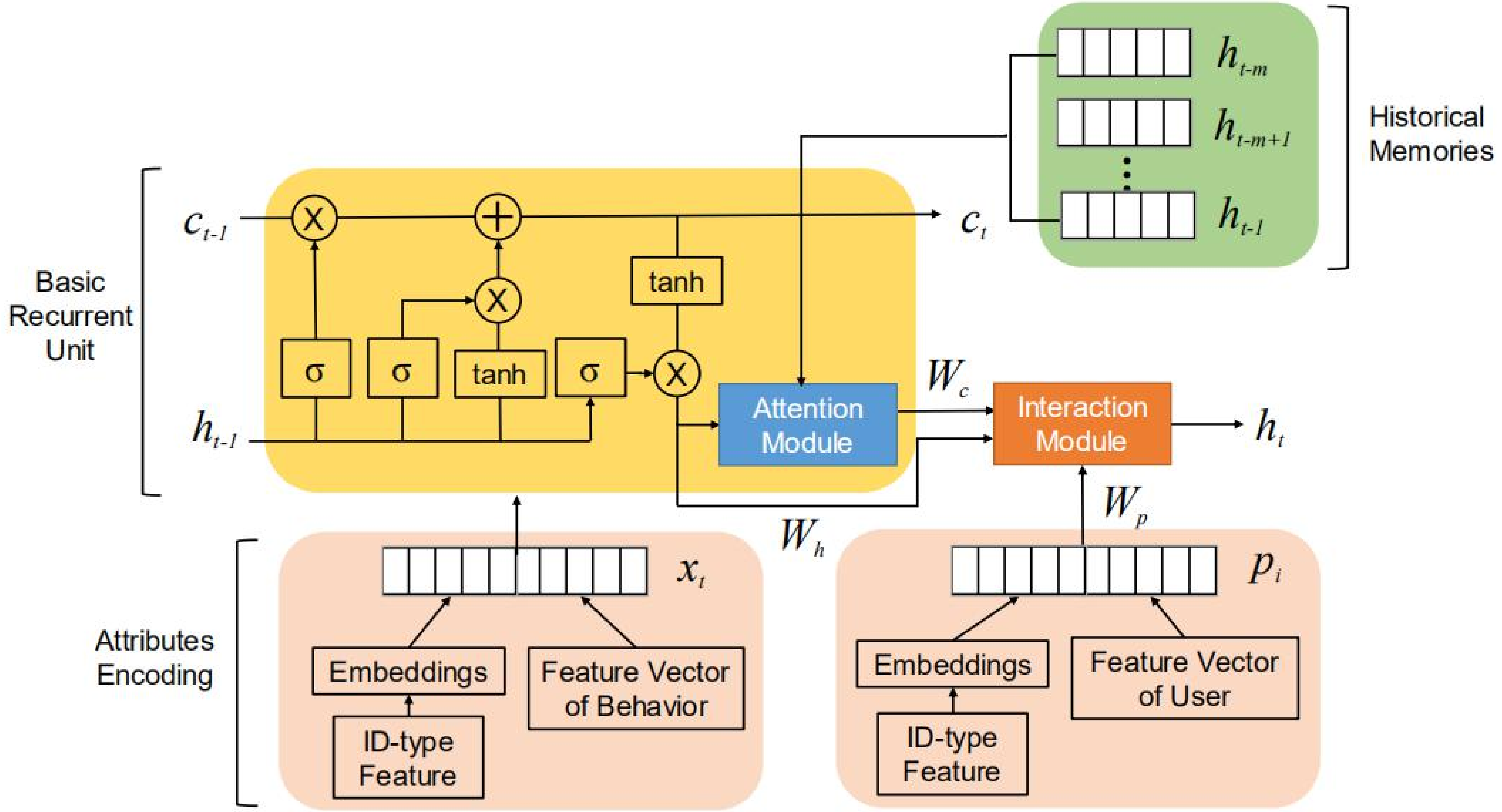
# Learning Sequential Behavior Representations for Fraud Detection

Jia Guo, Guannan Liu*, Yuan Zuo, Junjie Wu
*School of Economics and Management*
*Beihang University*
Beijing, China
{guojia1608,liugn,zuoyuan,wujj}@buaa.edu.cn

**ICDM-18**

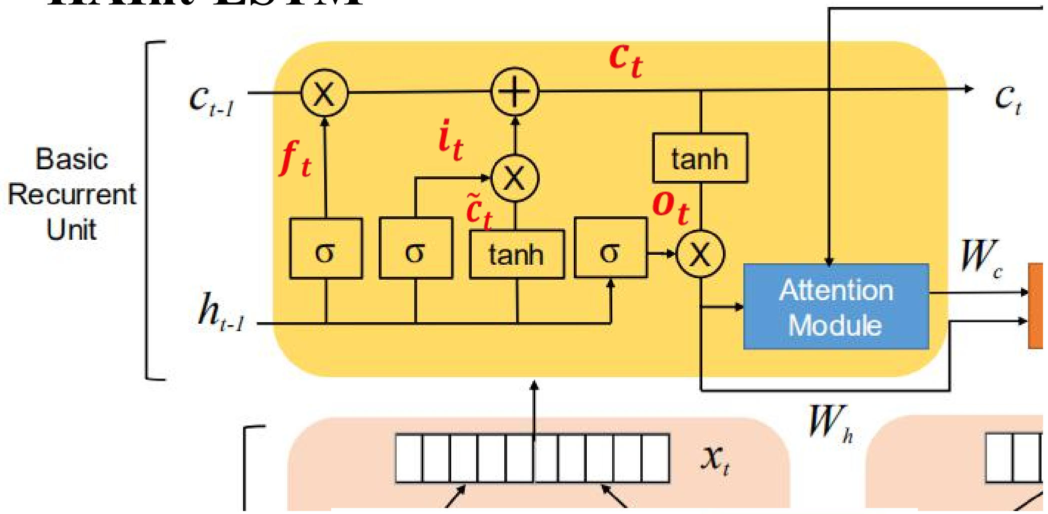
# Learning Sequential Behavior Representations for Fraud Detection

## HAInt-LSTM

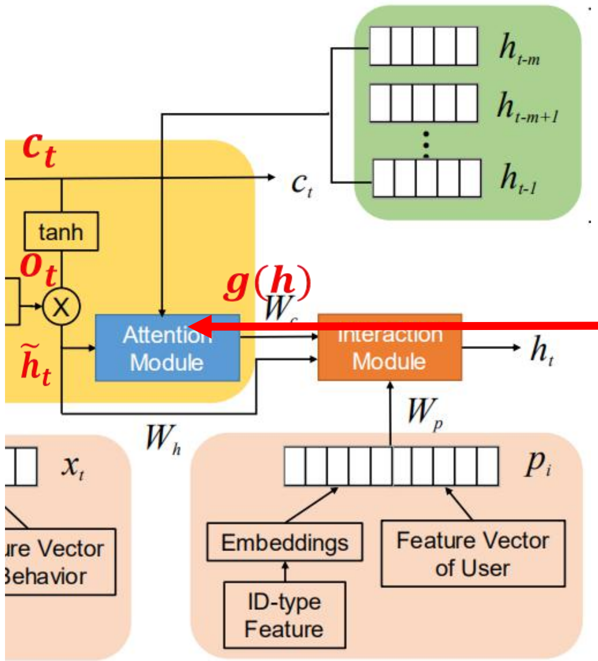


# Learning Sequential Behavior Representations for Fraud Detection

## HAInt-LSTM

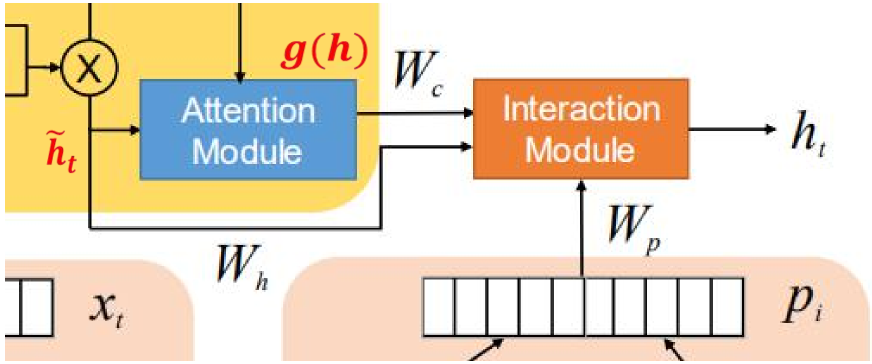



$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + W_{ft}\Delta T_{t-1,t} + b_f)$$
$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i)$$
$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o)$$
$$\tilde{c}_t = \tanh(W_{ch}h_{t-1} + W_{cx}x_t + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$




$$\tilde{h}_t = o_t \odot \tanh(c_t)$$
$$e_{t,k} = W_\alpha \tanh(W_{\alpha s}s_t + W_{\alpha k}h_k + b_\alpha)$$
$$\alpha_{t,k} = softmax(e_{t,k})$$
$$g(h) = \sum_{k=t-m}^{t-1} \alpha_{t,k} \odot h_k$$

## HAInt-LSTM




$$h_t = tanh(W_h \tilde{h}_t + W_p p_i + W_c g(h))$$

# Learning Sequential Behavior Representations for Fraud Detection

**Unsupervised Learning:** In this framework, we detect fraudsters based on unsupervised sequence prediction. At each time step, we predict the caller's next target. Since we assume that normal users have some potential regularities inside their behavioral sequences, thus their sequence would be more easily predicted, and the loss value of learning model would be very small. However, if a sequence belongs to a fraudster who doesn't have a stable social connection and always changes targets, the loss value of model would rise. Specifically, the loss function of this framework is as follows:

$$y'_t = \sigma(W_t h_t + b_t), \tag{11}$$

$$L_q = -\frac{1}{n}\frac{1}{T}\sum_n \sum_T [y_t \ln(y'_t) + (1 - y_t)\ln(1 - y'_t)], \tag{12}$$

where $y_t$ is the true target of $(t+1)$-th time step, $y'_t$ is the predicted target for $(t+1)$-th time step. $T$ is the maximum sequence length, $n$ is the total number of sequences.

**Supervised Learning:** In this framework, we assume that there is a small set of ground truth data, then we can build a classifier based on users' consecutive behaviors to predict their labels. We add a soft-max layer on the last sequence representation, so that utilizing the recurrent neural network as a classifier and give a possible label for this sequence. The loss function of this framework is as follows:

$$y' = \sigma(W_T h_T + b_T), \tag{13}$$

$$L_u = -\frac{1}{n}\sum_n [y \ln(y') + (1 - y)\ln(1 - y')], \tag{14}$$

where $y$ is the true label of the sequence, $y'$ is the predicted label of the sequence, $T$ is the maximum sequence length, $n$ is the total number of sequences.

# NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks

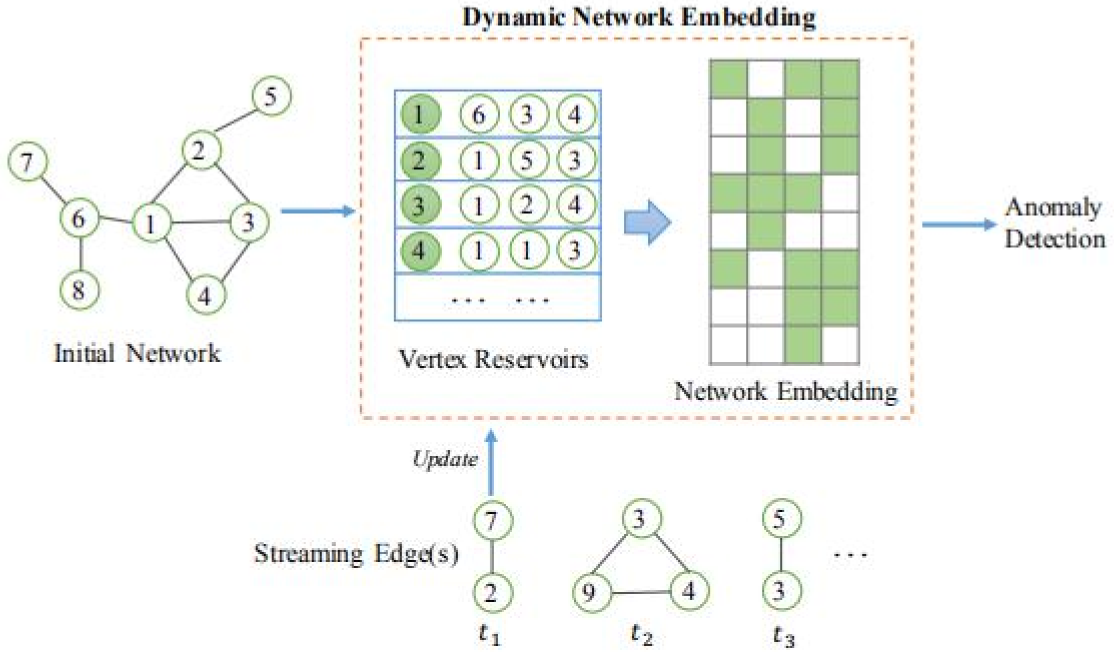
## NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks

Wenchao Yu[1], Wei Cheng[2], Charu C. Aggarwal[3], Kai Zhang[4], Haifeng Chen[2], and Wei Wang[1]
[1]Department of Computer Science, University of California Los Angeles
[2]NEC Laboratories America, Inc. [3]IBM Research AI
[4]Department of Computer and Information Sciences, Temple University
{wenchaoyu,weiwang}@cs.ucla.edu,charu@us.ibm.com,{weicheng,haifeng}@nec-labs.com,zhang.kai@temple.edu

**KDD-18**

# NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks

动态属性图的异常检测

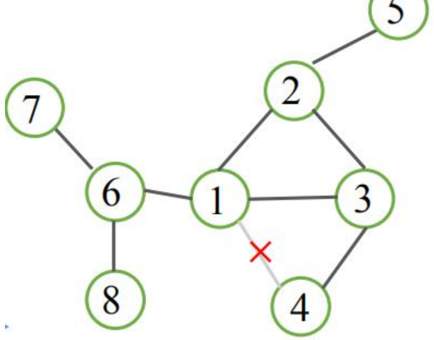



The proposed NetWalk is flexible. It is applicable on both **directed** and **undirected** networks, either weighted or not, to detect abnormal **vertices** and **edges** in a network that may evolve over time by dynamically inserting or deleting vertices and edges.

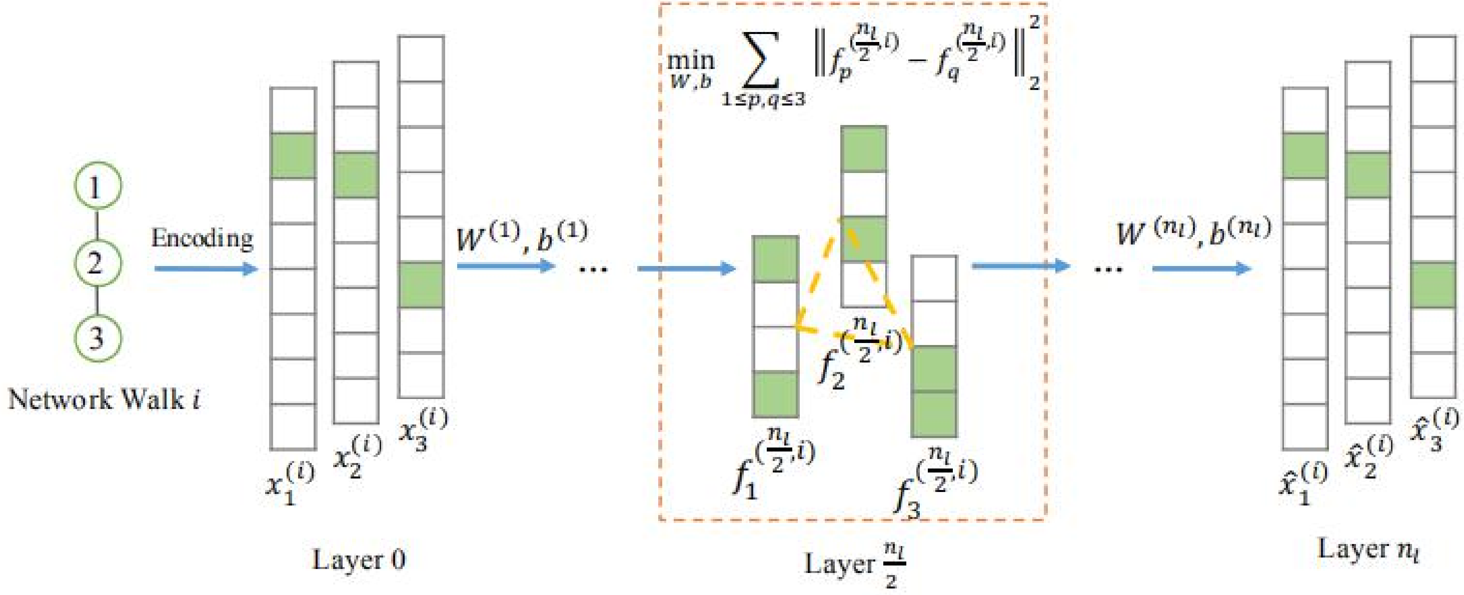
# NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks

## Network Walk

*Definition 3.1 (Network Walk).* For a given vertex $v_1 \in \mathcal{V}$ in a network $\mathcal{G}(\mathcal{E}, \mathcal{V})$, its network walk set is defined as $\Omega_{v_1} = \{(v_1, v_2, ..., v_l) | \ (v_i, v_{i+1}) \in \mathcal{E} \wedge p(v_i, v_{i+1}) = \frac{1}{D_{v_i, v_i}}\}$, which is a collection of $l$-hop walks starting from vertex $v_1$. The transition probability $p(v_i, v_{i+1})$ from $v_i$ to $v_{i+1}$ is proportional to the degree $D_{v_i, v_i}$ of vertex $v_i$. We call $\Omega_v$ a network walk set starting from $v$, and $\Omega = \{\Omega_v\}_{v \in \mathcal{V}}$ as the union of all walks.




| | | | |
|---|---|---|---|
| 1 | 6 | 3 | 6 |
| 2 | 1 | 5 | 1 |
| 3 | 1 | 4 | 4 |
| 4 | 3 | 3 | 3 |
| … | … | | |



# NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks

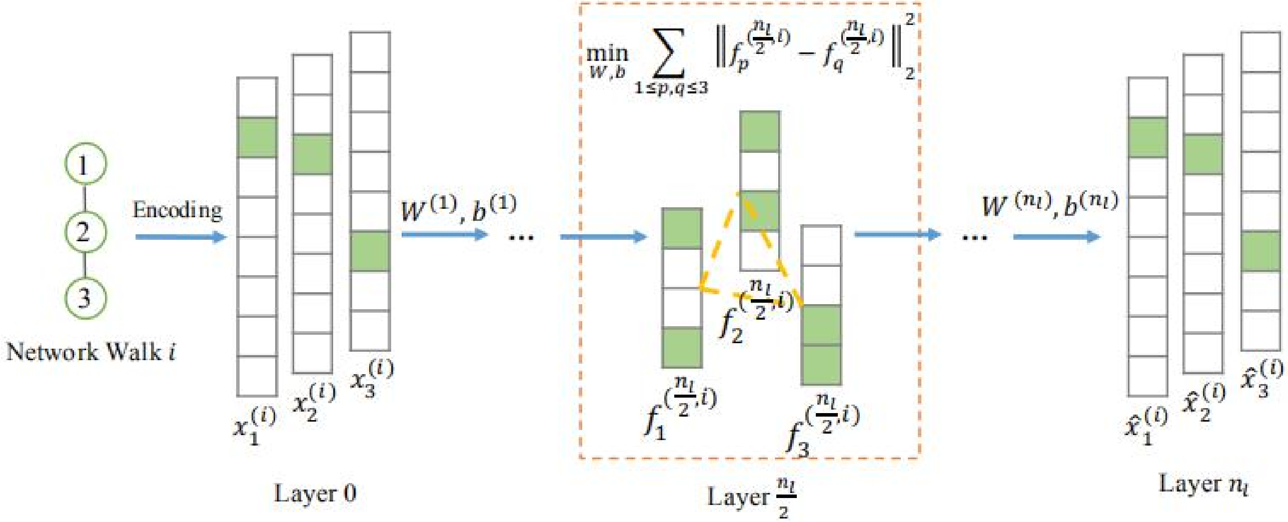
## Learning Network Representations

Formally, given a one-hot encoded network walk $\{\mathbf{x}_p^{(i)}\}_{p=1}^{l}$, $i = 1, \ldots, |\Omega|$, we want to learn the following representations in a $n_l$-layer autoencoder network,

$$f^{(\frac{n_l}{2})}(\mathbf{x}_p^{(i)}) = \sigma(\mathbf{W}^{(\frac{n_l}{2})\top} h^{(\frac{n_l}{2})}(\mathbf{x}_p^{(i)}) + \mathbf{b}^{(\frac{n_l}{2})}), \qquad (1)$$

where

$$h^{(\frac{n_l}{2})}(\mathbf{x}_p^{(i)}) = \mathbf{W}^{(\frac{n_l}{2}-1)} f^{(\frac{n_l}{2}-1)}(\mathbf{x}_p^{(i)}) + \mathbf{b}^{(\frac{n_l}{2}-1)}. \qquad (2)$$




$$J_{AE} = \frac{1}{2} \sum_{i=1}^{|\Omega|} \sum_{p=1}^{l} \left\| f^{(n_l)}(\mathbf{x}_p^{(i)}) - \mathbf{x}_p^{(i)} \right\|_2^2.$$

$$J_{Clique} = \sum_{i=1}^{|\Omega|} \sum_{1\le p,q\le l} \left\| f^{(\frac{n_l}{2})}(\mathbf{x}_p^{(i)}) - f^{(\frac{n_l}{2})}(\mathbf{x}_q^{(i)}) \right\|_2^2$$

Due to the sparsity of the input and output vectors, we consider a sparse auto-encoder with sparsity parameter $\rho$ and penalize it with the Kullback-Leibler divergence [27],

$$\mathrm{KL}(\rho\|\hat{\boldsymbol{\rho}}^{(\ell)}) = \sum_{j=1}^{d} \mathrm{KL}(\rho\|\hat{\rho}_j^{(\ell)}) = \sum_{j=1}^{d} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}, \qquad (5)$$

$$\hat{\boldsymbol{\rho}}^{(\ell)} = \frac{1}{|\Omega|\times l} \sum_{i=1}^{|\Omega|} \sum_{p=1}^{l} f^{(\ell)}(\mathbf{x}_p^{(i)})$$

$$J(\mathbf{W}, \mathbf{b}) = \underbrace{\sum_{i=1}^{|\Omega|} \sum_{1\le p,q\le l} \left\| f^{(\frac{n_l}{2})}(\mathbf{x}_p^{(i)}) - f^{(\frac{n_l}{2})}(\mathbf{x}_q^{(i)}) \right\|_2^2}_{\text{Clique Embedding Loss}} + \underbrace{\frac{\gamma}{2} \sum_{i=1}^{|\Omega|} \sum_{p=1}^{l} \left\| f^{(n_l)}(\mathbf{x}_p^{(i)}) - \mathbf{x}_p^{(i)} \right\|_2^2}_{\text{Reconstruction Error}}$$
$$+ \underbrace{\beta \sum_{\ell=1}^{n_l-1} \sum_{j} \mathrm{KL}\left(\rho\|\hat{\rho}_j^{(\ell)}\right)}_{\text{Sparsity Constraint}} + \underbrace{\frac{\lambda}{2} \sum_{\ell=1}^{n_l} \left\|\mathbf{W}^{(\ell)}\right\|_F^2}_{\text{Weight Decay}}, \qquad (6)$$

# NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks

## Objective Function:

$$J(\mathbf{W},\mathbf{b}) = \underbrace{\sum_{i=1}^{|\Omega|} \sum_{1\le p,q\le l} \left\| f^{(\frac{n_l}{2})}(\mathbf{x}_p^{(i)}) - f^{(\frac{n_l}{2})}(\mathbf{x}_q^{(i)}) \right\|_2^2}_{\text{Clique Embedding Loss}} + \underbrace{\frac{\gamma}{2} \sum_{i=1}^{|\Omega|} \sum_{p=1}^{l} \left\| f^{(n_l)}(\mathbf{x}_p^{(i)}) - \mathbf{x}_p^{(i)} \right\|_2^2}_{\text{Reconstruction Error}}$$

$$+ \underbrace{\beta \sum_{\ell=1}^{n_l-1} \sum_j \mathrm{KL}\left(\rho \| \hat{\rho}_j^{(\ell)}\right)}_{\text{Sparsity Constraint}} + \underbrace{\frac{\lambda}{2} \sum_{\ell=1}^{n_l} \left\| \mathbf{W}^{(\ell)} \right\|_F^2}_{\text{Weight Decay}}, \qquad (6)$$

The loss function $J(\mathbf{W}, \mathbf{b})$ can also be written in a matrix form:

$$J(\mathbf{W},\mathbf{b}) = \sum_{i=1}^{|\Omega|} \mathrm{Tr}(\mathcal{F}^{(i)} \mathbf{L} \mathcal{F}^{(i)\top}) + \frac{\gamma}{2} \left\| \mathcal{H}^{(n_l)}(\mathbf{X}) - \mathbf{X} \right\|_F^2$$
$$+ \beta \sum_{\ell=1}^{n_l-1} \mathrm{KL}(\rho \| \hat{\boldsymbol{\rho}}^{(\ell)}) + \frac{\lambda}{2} \left\| \mathbf{W}^{(1)} \right\|_F^2 + \frac{\lambda}{2} \sum_{\ell=1}^{n_l} \left\| \mathbf{W}^{(\ell)} \right\|_F^2$$

where $\mathcal{F}^{(i)} = [f_1^{(i)}, f_2^{(i)}, \ldots, f_l^{(i)}]$, $f_l^{(i)} = f^{(\frac{n_l}{2})}(\mathbf{x}_l^{(i)})$; $\mathbf{L}$ is the Laplacian matrix of the clique with $l$ vertices, thus we have $\mathbf{L} = \mathbf{I}_l \times (l-1) - \Phi$, and $\Phi_{i,j} = 1, \forall i \neq j$. $\mathbf{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(|\Omega|)}]$, $\mathbf{x}^{(i)} = [\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \ldots, \mathbf{x}_l^{(i)}]$; $\mathcal{H}^{(n_l)}(\mathbf{X}) = [g^{(1)}, g^{(2)}, \ldots, g^{(|\Omega|)}]$, $g^{(i)} = [f^{(n_l)}(\mathbf{x}_1^{(i)}), f^{(n_l)}(\mathbf{x}_2^{(i)}), \ldots f^{(n_l)}(\mathbf{x}_l^{(i)})]$.

# NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks

$n > \frac{l}{2}$:

$$\nabla_{\mathbf{W}^{(\ell)}} J(\mathbf{W}, \mathbf{b}) = \delta^{(\ell)} \left(f^{(\ell-1)}(\mathbf{X})\right)^\top + \lambda \mathbf{W}^{(\ell)},$$

$$\nabla_{\mathbf{b}^{(\ell)}} J(\mathbf{W}, \mathbf{b}) = \sum_{i=1}^{|\Omega|} \delta_i^{(\ell)}.$$

*else*:

$$\nabla_{\mathbf{W}^{(\ell)}} J(\mathbf{W}, \mathbf{b}) = \sum_{i=1}^{|\Omega|} \mathcal{F}^{(i)} (\mathbf{L} + \mathbf{L}^\top) \circ \mathcal{F}^{(i)} \circ (1 - \mathcal{F}^{(i)}) \left(f^{(\ell-1)}(\mathbf{X})\right)^\top + \delta^{(\ell)} \left(f^{(\ell-1)}(\mathbf{X})\right)^\top + \lambda \mathbf{W}^{(\ell)},$$

$$\nabla_{\mathbf{b}^{(\ell)}} J(\mathbf{W}, \mathbf{b}) = \sum_{i=1}^{|\Omega|} \mathcal{F}^{(i)} (\mathbf{L} + \mathbf{L}^\top) \circ \mathcal{F}^{(i)} \circ (1 - \mathcal{F}^{(i)}) + \delta_i^{(\ell)}.$$

**Algorithm 1:** Clique Embedding of NetWalk

**Input:** Network walk set $\Omega$.

**Output:** Network representations $f^{\frac{n_l}{2}}(\mathbf{x}_p^{(i)})$

Set latent dimension $d$, sparsity $\rho$, weight control parameters $\gamma$, $\beta$ and $\lambda$.

Randomly initialize $\{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^{n_l}$.

Construct input vector $\mathbf{x}_p^{(i)} \in \mathbb{R}^n$ for vertex $p$ in walk $i$, $1 \le p \le l$, $1 \le i \le |\Omega|$

**while** *not stopping criterion* **do**

- Perform a feedforward pass to compute $f^{(\ell)}(\mathbf{x}_p^{(i)})$.
- For the output layer $n_l$, set $\delta^{(n_l)}$ using Eq.(8)
- **for** $\ell = n_l - 1, n_l - 2, n_l - 3, \ldots, 1$ **do**
  - Compute "error terms" $\delta(\ell)$ using Eq.(9).
  - **if** $\ell > \frac{n_l}{2}$ **then**
    - Compute partial derivatives $\nabla_{\mathbf{W}^{(\ell)}} J(\mathbf{W}, \mathbf{b})$ and $\nabla_{\mathbf{b}^{(\ell)}} J(\mathbf{W}, \mathbf{b})$ using Eq.(10)-(11).
  - **else**
    - Compute partial derivatives $\nabla_{\mathbf{W}^{(\ell)}} J(\mathbf{W}, \mathbf{b})$ and $\nabla_{\mathbf{b}^{(\ell)}} J(\mathbf{W}, \mathbf{b})$ using Eq.(12)-(13).
  - Determine the step size $\xi$ by line search.
  - Update $\mathbf{W}^{(\ell)} = \mathbf{W}^{(\ell)} - \xi \nabla_{\mathbf{W}^{(\ell)}} J(\mathbf{W}, \mathbf{b})$.
  - Update $\mathbf{b}^{(\ell)} = \mathbf{b}^{(\ell)} - \xi \nabla_{\mathbf{b}^{(\ell)}} J(\mathbf{W}, \mathbf{b})$.

Compute embedding results $f^{\frac{n_l}{2}}(\mathbf{x}_p^{(i)})$.

# NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks

## Edge Encoding:

$(v, u)$, the edge representation should be the same. In this paper, we use the Hadamard operator which has shown good performance in edge encoding [15]. Assume that the $d$-dimensional representation learned by Algorithm 1 for vertex $v$ is $f(v)$, then the representation of each edge $(v, u)$ under Hadamard operator is $[f(v) \circ f(u)]_i = f_i(v) \times f_i(u)$. It is worth mentioning that the way to encode edges is very flexible. We can add any additional edge-specific features to augment the edge vector.

# NetWalk: A Flexible Deep Embedding Approach for Anomaly Detection in Dynamic Networks

## Maintaining Network Representations Incrementally(处理结点和边的增减):

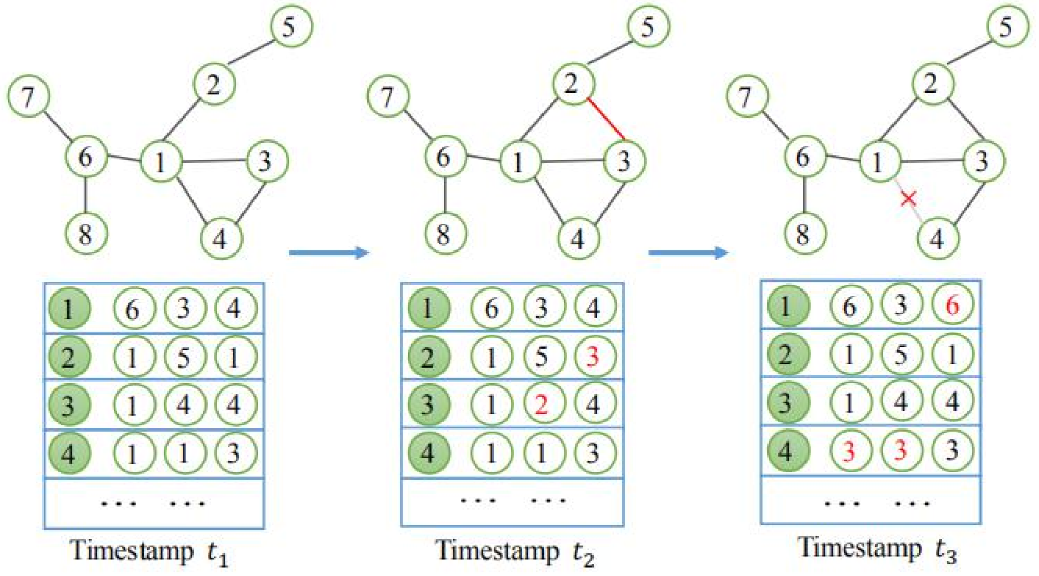



**Figure 4: Illustration of updating the reservoirs. Initially we build the reservoir of each vertex based on the network at $t_1$. When $(v_2, v_3)$ is added at timestamp $t_2$, the corresponding reservoirs of $v_2$ and $v_3$ will be updated. Similarly, when $(v_1, v_4)$ is deleted at timestamp $t_3$, we replace the deleted items with the remaining neighbors of the corresponding vertex.**

as new edges arrive. The updating rules are described as follows for each newly added edge $(u, v)$:

(1) update the degree of vertices $u$ and $v$: $D_{u,u} = D_{u,u} + 1$, $D_{v,v} = D_{v,v} + 1$;

(2) for each item in the reservoir $S_u$, with probability $\frac{1}{D_{u,u}}$, replace the old item with the new item $v$; and with probability $1 - \frac{1}{D_{u,u}}$, keep the old item;

(3) for each item in the reservoir $S_v$, with probability $\frac{1}{D_{v,v}}$, replace the old item with the new item $u$; and with probability $1 - \frac{1}{D_{v,v}}$, keep the old item.

In case where edges are deleted, the reservoir is chosen similarly to aforementioned rules.

## ANOMALY DETECTION

When new edges stream in, we need to update cluster centers accordingly. In this paper, we leverage the streaming k-means clustering [4] which uses parameters to control the decay of estimates.

in our model, we find the closest cluster to each point. We use the Euclidean distance as the similarity measure, given by $||\boldsymbol{c} - f(\cdot)||_2$, where $\boldsymbol{c}$ is the cluster center and $f(\cdot)$ is the learned representation for each vertex or edge. The anomaly score for each point is reported as its closest distance to any cluster centers.

Assuming that there are $n_0$ points $\{\boldsymbol{x}_i\}_{i=1}^{n_0}$ in an existing cluster and $n'$ new points $\{\boldsymbol{x}'_i\}_{i=1}^{n'}$ at time-stamp $T'$ to be absorbed by this cluster, the centroid $\boldsymbol{c}$ can be updated in the following way

$$\boldsymbol{c} = \frac{\alpha \boldsymbol{c}_0 n_0 + (1-\alpha)\sum_{i=1}^{n'} \boldsymbol{x}'_i}{\alpha n_0 + (1-\alpha) n'}, \tag{15}$$

where $\boldsymbol{c}_0$ is the previous cluster center. The decay factor $\alpha$ is chosen as 0.5 and used to ignore older instances, which is analogous to an exponentially-weighted moving average.

## On the weighted networks

**First**, since the walks generating step adopts random walker technique, it is easy to consider the weights of edges into the transition probability.

**Accordingly**, in Eq.(4), additional weights should be put to the pairwise loss of two vertices.

$$J_{Clique} = \sum_{i=1}^{|\Omega|} \sum_{1 \le p, q \le l} \left\| f^{(\frac{n_l}{2})}(\mathbf{x}_p^{(i)}) - f^{(\frac{n_l}{2})}(\mathbf{x}_q^{(i)}) \right\|_2^2$$

# Cash-Out User Detection Based on Attributed Heterogeneous Information Network with a Hierarchical Attention Mechanism

Cash-Out User Detection Based on Attributed Heterogeneous Information Network with a Hierarchical Attention Mechanism

**Binbin Hu,[1] Zhiqiang Zhang,[2] Chuan Shi,[1] Jun Zhou,[2] Xiaolong Li,[2] Yuan Qi[2]**
[1] Beijing University of Posts and Telecommunications
[2] AI Department, Ant Financial Services Group
{hubinbin, shichuan}@bupt.edu.cn, {lingyao.zzq, jun.zhoujun, xl.li, yuan.qi}@antfin.com

**BUPT & 蚂蚁金服**

**AAAI-19**

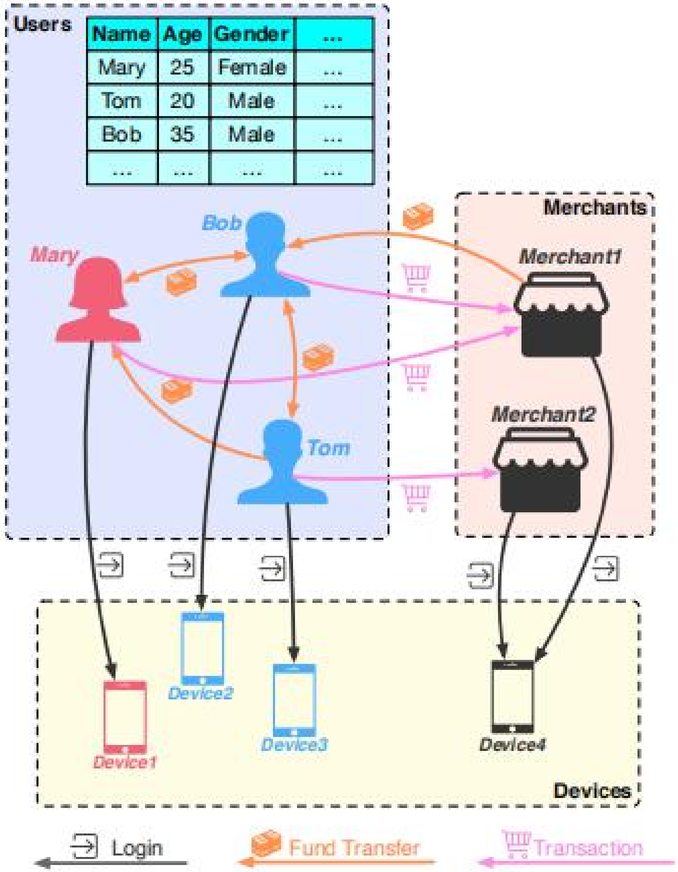
# Cash-Out User Detection Based on Attributed Heterogeneous Information Network with a Hierarchical Attention Mechanism

**Motivation**:

传统的**挖掘套现用户**的方式是建模为分类问题，然后手工构造大量的静态特征。然而在实际场景种用户之间存在丰富的交互关系，这种关系对于发现套现用户是有帮助的。

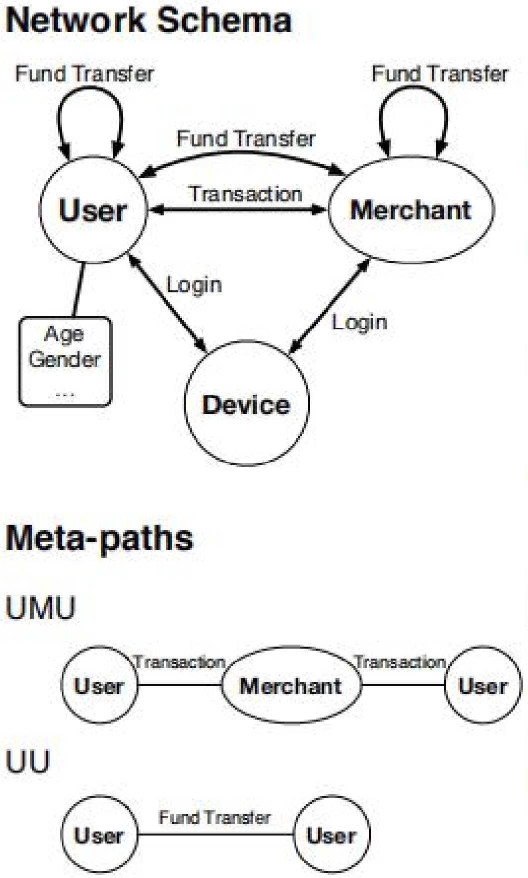
# Cash-Out User Detection Based on Attributed Heterogeneous Information Network with a Hierarchical Attention Mechanism

## Attributed Heterogeneous Information Network (AHIN)




| Name | Age | Gender | ... |
|---|---|---|---|
| Mary | 25 | Female | ... |
| Tom | 20 | Male | ... |
| Bob | 35 | Male | ... |
| ... | ... | ... | ... |

(a) Scenario of credit payment service

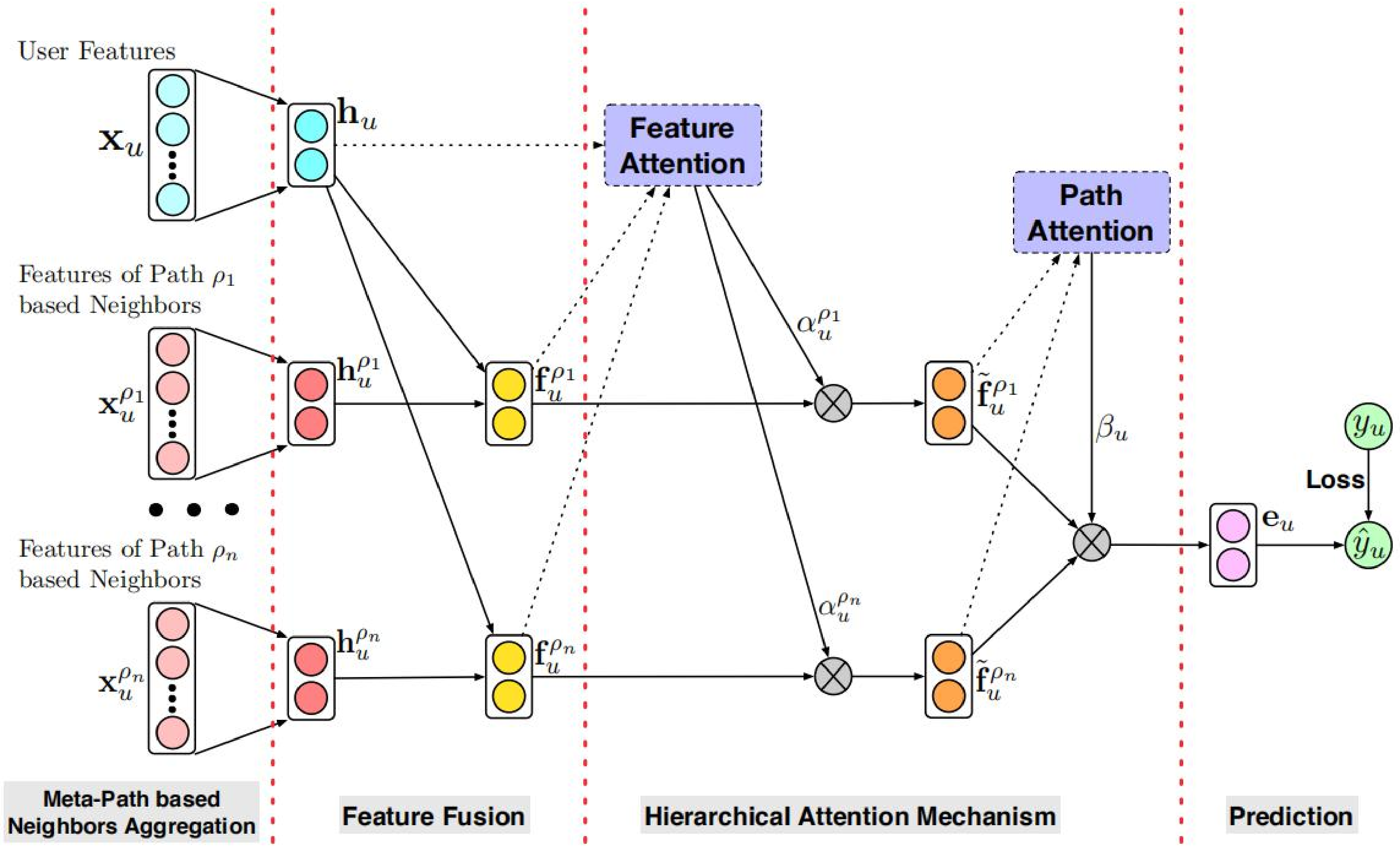



(b) Network schema and meta-path examples

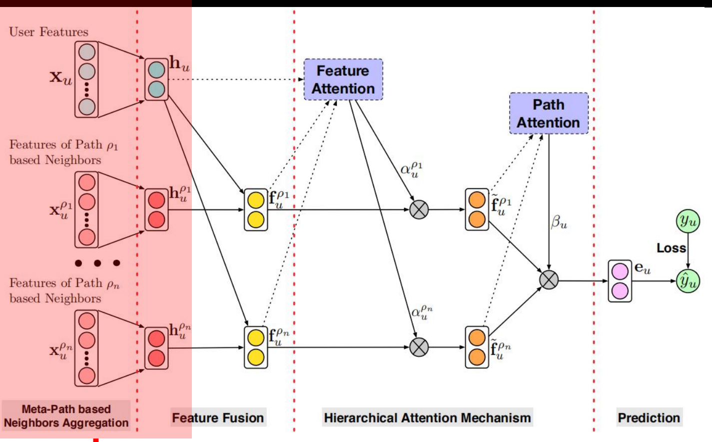
**Definition 2** ***Meta-path*** *(Sun et al. 2011). A meta-path $\rho$ is defined as a path in the form of $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$ (abbreviated as $A_1 A_2 \cdots A_{l+1}$), which describes a composite relation $R = R_1 \circ R_2 \circ \cdots \circ R_l$ between object $A_1$ and $A_{l+1}$, where $\circ$ denotes the composition operator on relations.*

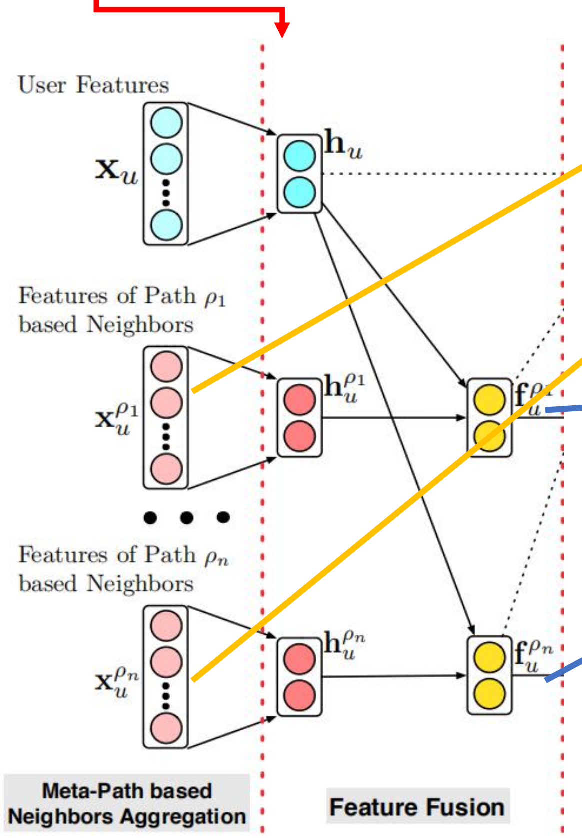
**Definition 3** ***Meta-path based Neighbors****. Giving a user $u$ in an AHIN, the meta-path based neighbors is defined as the set of aggregate neighbors under the given meta-path for the user $u$ in the AHIN.*

# Cash-Out User Detection Based on Attributed Heterogeneous Information Network with a Hierarchical Attention Mechanism



# Cash-Out User Detection Based on Attributed Heterogeneous Information Network with a Hierarchical Attention Mechanism




$$\mathbf{x}_u^{\rho} = \sum_{j \in \mathcal{N}_u^{\rho}} w_{uj}^{\rho} * \mathbf{x}_j$$

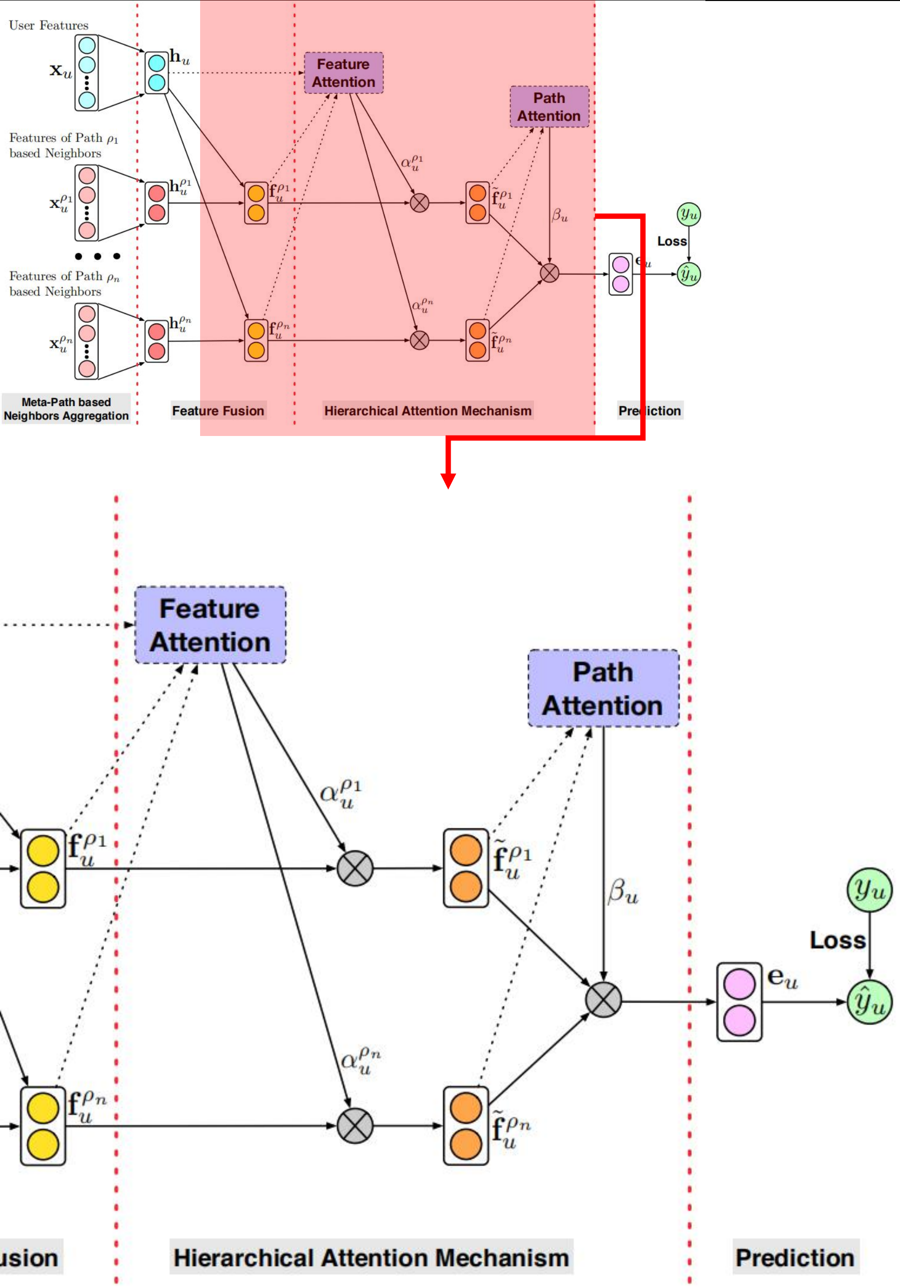
where $\mathcal{N}_u^{\rho}$ is the neighbors of node $j$ based on meta-path $\rho$ and $\mathbf{x}_j$ represents the attribute information vector associated with node $j$. The given link weight $w_{uj} > 0$ for weighted networks and $w_{uj} = 1$ for unweighted networks.

$$\mathbf{h}_u = \mathbf{W}\mathbf{x}_u + \mathbf{b}, \quad \mathbf{h}_u^{\rho} = \mathbf{W}^{\rho}\mathbf{x}_u^{\rho} + \mathbf{b}^{\rho}$$

$$\mathbf{f}_u^{\rho} = \mathrm{ReLU}(\mathbf{W}_F^{\rho} g(\mathbf{h}_u, \mathbf{h}_u^{\rho}) + \mathbf{b}_F^{\rho})$$

$g(\cdot, \cdot)$ is the fusion function, which can be concatenation, addition or element-wise product.

# Cash-Out User Detection Based on Attributed Heterogeneous Information Network with a Hierarchical Attention Mechanism




## Feature Attention:

$$\begin{aligned} \boldsymbol{v}_u^\rho &= \mathrm{ReLU}(\mathbf{W}_f^1[\mathbf{h}_u; \mathbf{f}_u^\rho] + \mathbf{b}_f^1), \\ \boldsymbol{\alpha}_u^\rho &= \mathrm{ReLU}(\mathbf{W}_f^2 \boldsymbol{v}_u^\rho + \mathbf{b}_f^2), \end{aligned}$$

$$\hat{\alpha}_{u,i}^\rho = \frac{\exp(\alpha_{u,i}^\rho)}{\sum_{j=1}^K \exp(\alpha_{u,j}^\rho)}$$

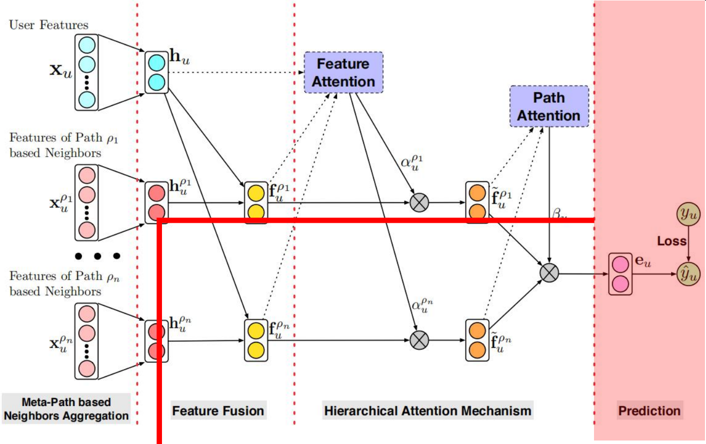
$$\widetilde{\mathbf{f}}_u^\rho = \hat{\boldsymbol{\alpha}}_u^\rho \bigodot \mathbf{f}_u^\rho$$

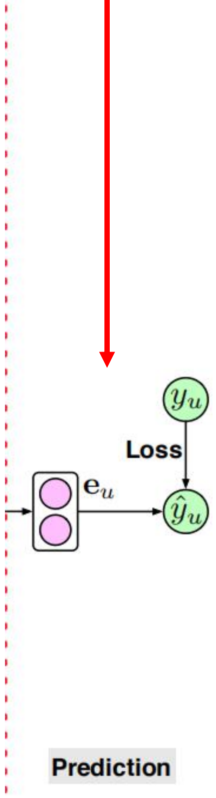
## Path Attention:

$$\beta_{u,\rho} = \frac{\exp(\mathbf{z}^{\rho\,\mathrm{T}} \cdot \widetilde{\mathbf{f}}_u^C)}{\sum_{\rho' \in \mathcal{P}} \exp(\mathbf{z}^{\rho'\,\mathrm{T}} \cdot \widetilde{\mathbf{f}}_u^C)}$$

$$\mathbf{e}_u = \sum_{\rho \in \mathcal{P}} \beta_{u,\rho} * \widetilde{\mathbf{f}}_u^\rho$$

# Cash-Out User Detection Based on Attributed Heterogeneous Information Network with a Hierarchical Attention Mechanism




## Model Learning:

$$\mathbf{z}_u = \mathrm{ReLU}(\mathbf{W}_L \cdots \mathrm{ReLU}(\mathbf{W}_1 \mathbf{e}_u + \mathbf{b}_1) + \mathbf{b}_L)$$

$$p_u = \mathrm{sigmoid}(\mathbf{w}_p^T \mathbf{z}_u + b_p).$$

## Optimization:

$$\mathcal{L}(\Theta) = \sum_{\langle u, y_u \rangle \in \mathcal{D}} (y_u \log(p_u) + (1 - y_u) \log(1 - p_u)) + \lambda ||\Theta||_2^2, \quad (12)$$

# Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework

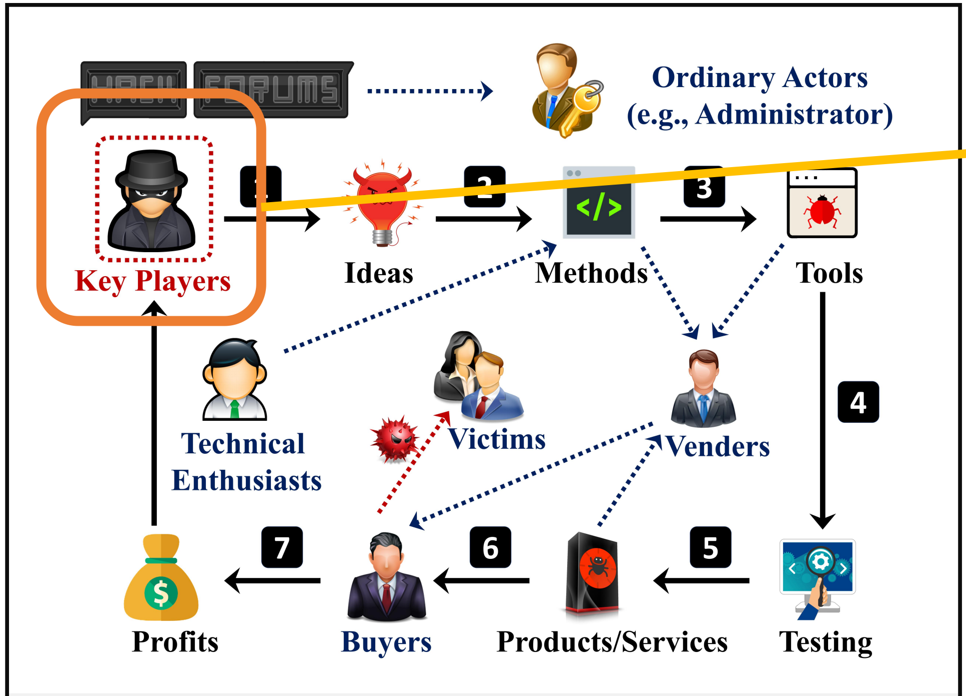
Yiming Zhang, Yujie Fan, Yanfang Ye*
Department of CDS, Case Western Reserve University, OH, USA

Liang Zhao
Department of IST
George Mason University
VA, USA

Chuan Shi
School of CS
Beijing University of Posts and Telecommunications, Beijing, China

**CIKM-19**

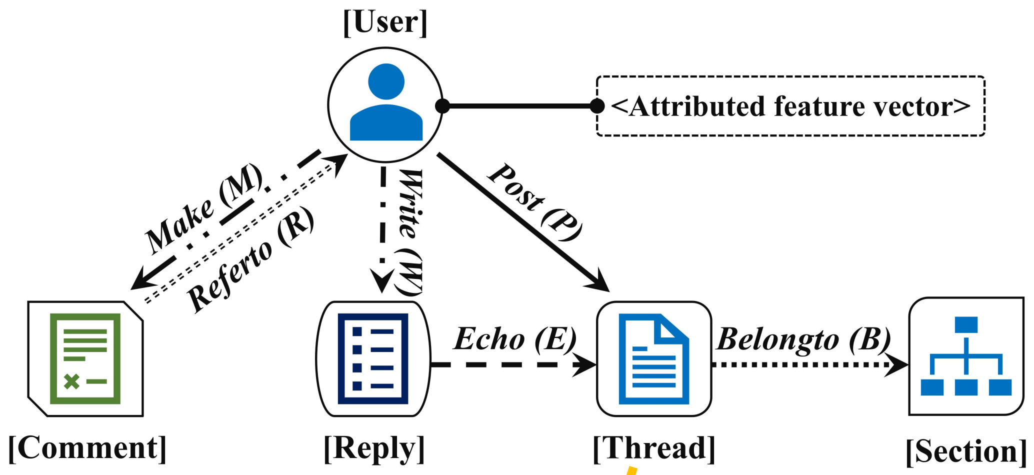
# Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework




**Role of key players in the value chain in Hack Forums**

1. Key players generate ideas by analysis of market demands
2. Exploit the vulnerabilities and devise the methods
3. Develop the tools using the devised methods
4. Testing the tools by purchasing some accounts and bots
5. Produce and advertise illicit products or services
6. Sell the illicit products or services
7. Monetize the illicit products or services

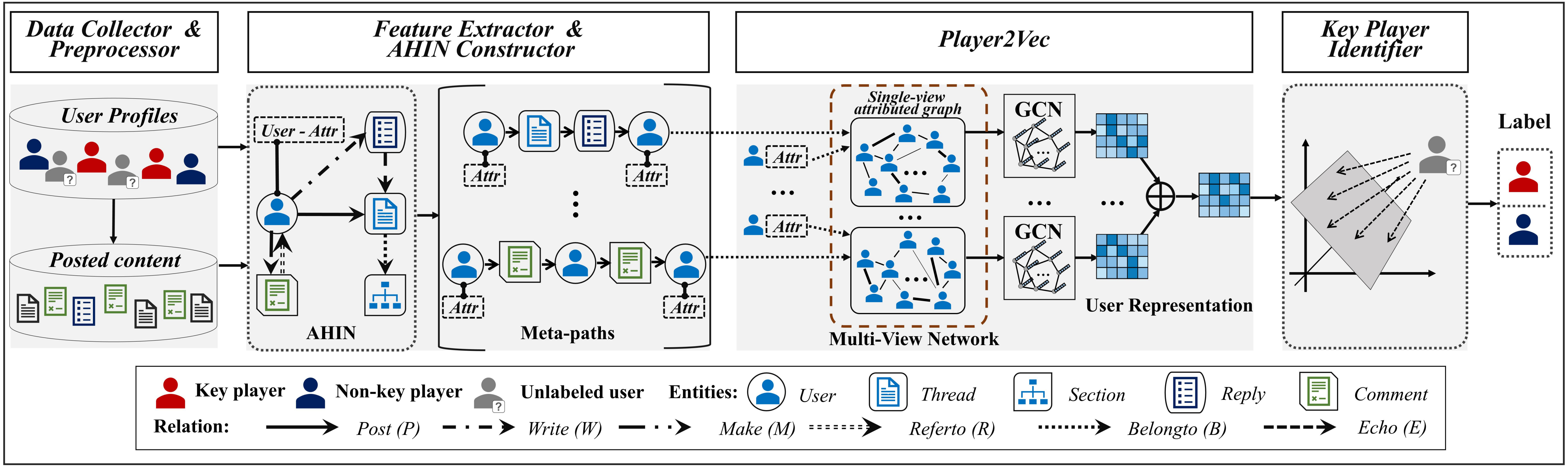
**Find out it!**




[可数名词](computing 计) a series of connected messages on a message board on the Internet which have been sent by different people （互联网留言板上帖子的）系列相关信息，链式消息

跟帖

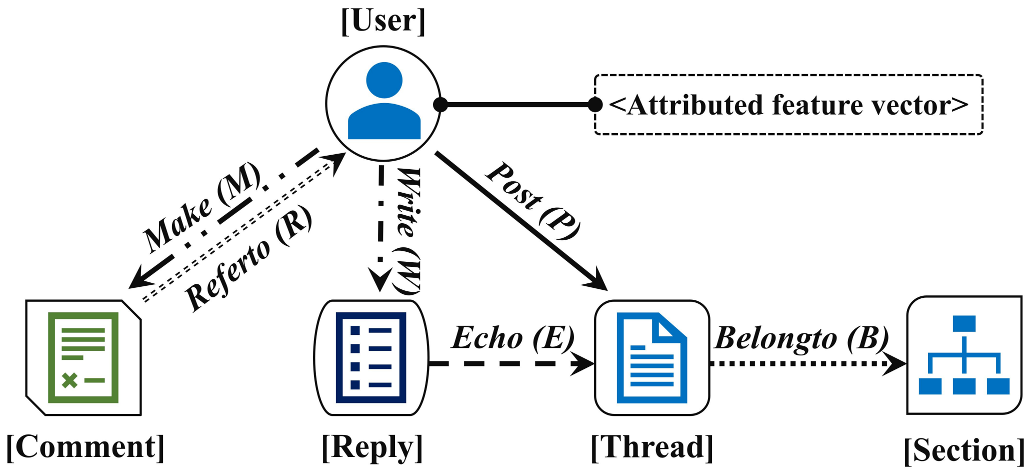
# Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework

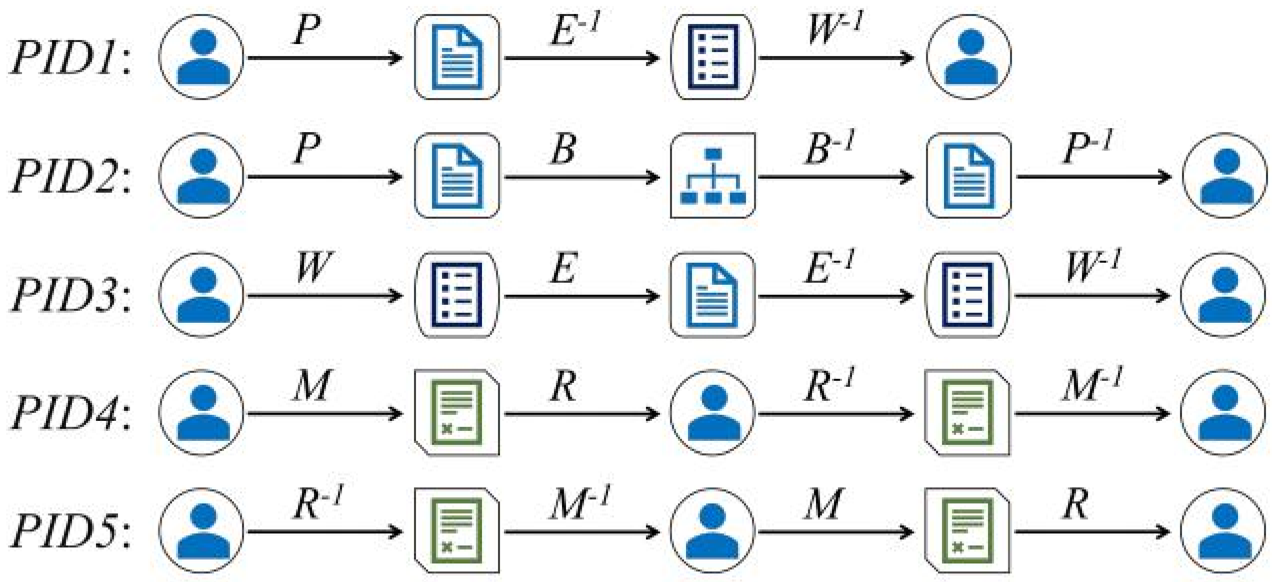


# Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework

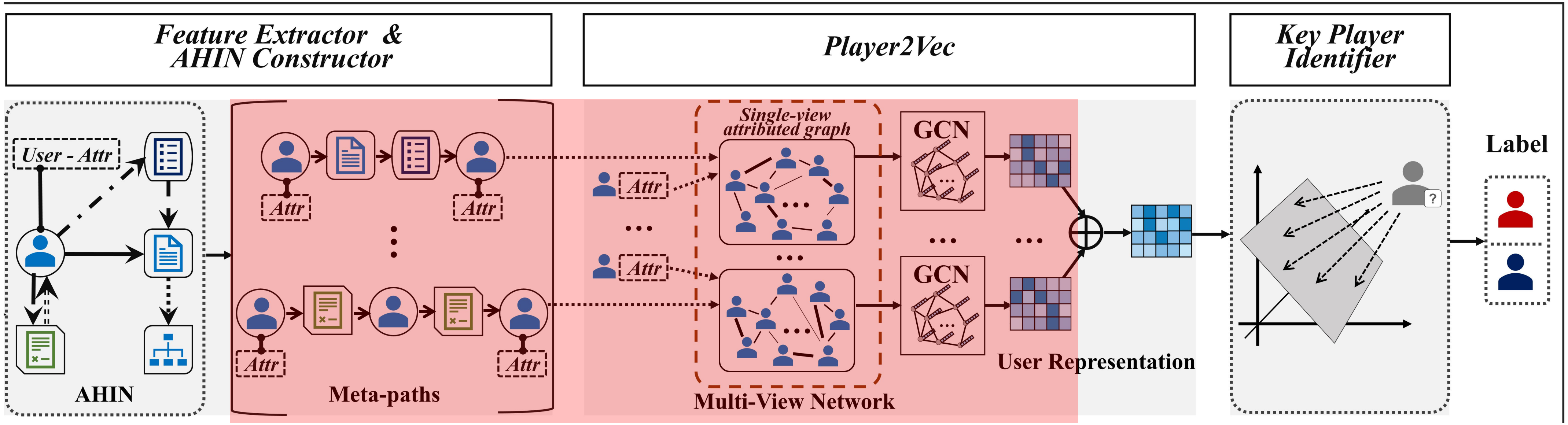
**Meta-Path:**

$$A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \dots \xrightarrow{R_L} A_{L+1}$$



# Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework



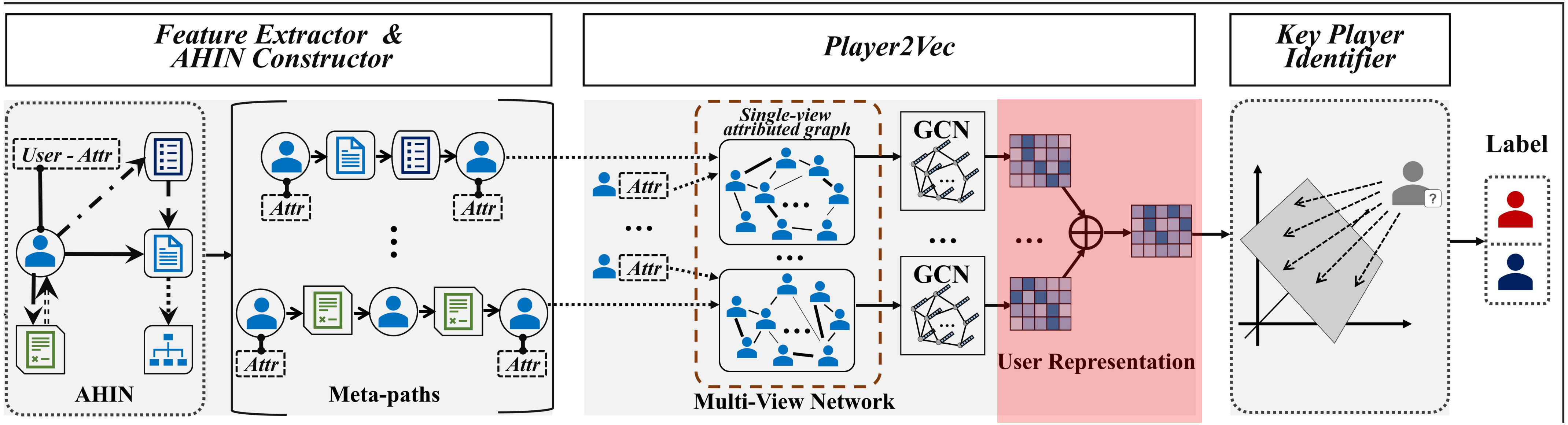

## Single-View Attributed Graph Embedding with GCN:

$$H^{k,l+1} = \sigma(\widetilde{A^k} H^{k,l} W^{k,l}).$$

$$\mathbf{f}^k = \widetilde{A^k}(\mathrm{ReLU}\ldots\mathrm{ReLU}(\widetilde{A^k} X W^{k,0})\ldots W^{k,L-2}) W^{k,L-1}).$$

$$\mathbf{f}^k = \mathrm{GCN}(\mathbf{X}, A^k) = \widetilde{A^k}\,\mathrm{ReLU}(\widetilde{A^k}\,\mathbf{X}\,W^{k,0})\,W^{k,1}.$$

# Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework




## Multi-View Network Embedding with Attention:

$$\alpha_{i,k} = \frac{\exp({\mathbf{z}^k}^T \cdot \mathbf{f}_i^C)}{\sum_{k' \in K} \exp({\mathbf{z}^{k'}}^T \cdot \mathbf{f}_i^C)}$$

$$\mathbf{e}_i = \sum_{k \in K} \alpha_{i,k} \cdot \mathbf{f}_i^k$$

Finally, we feed user embeddings to Support Vector Machine (**SVM**) to build the classification model for key player identification. In the experiments, we randomly select 90% of the data for training, while the remaining 10% is used for testing

# DEEP AUTOENCODING GAUSSIAN MIXTURE MODEL FOR UNSUPERVISED ANOMALY DETECTION

## DEEP AUTOENCODING GAUSSIAN MIXTURE MODEL FOR UNSUPERVISED ANOMALY DETECTION

**Bo Zong[†], Qi Song[‡], Martin Renqiang Min[†], Wei Cheng[†]**
**Cristian Lumezanu[†], Daeki Cho[†], Haifeng Chen[†]**
[†]NEC Laboratories America
[‡]Washington State University, Pullman
{bzong, renqiang, weicheng, lume, dkcho, haifeng}@nec-labs.com
qsong@eecs.wsu.edu

**ICLR-19**

# DEEP AUTOENCODING GAUSSIAN MIXTURE MODEL FOR UNSUPERVISED ANOMALY DETECTION

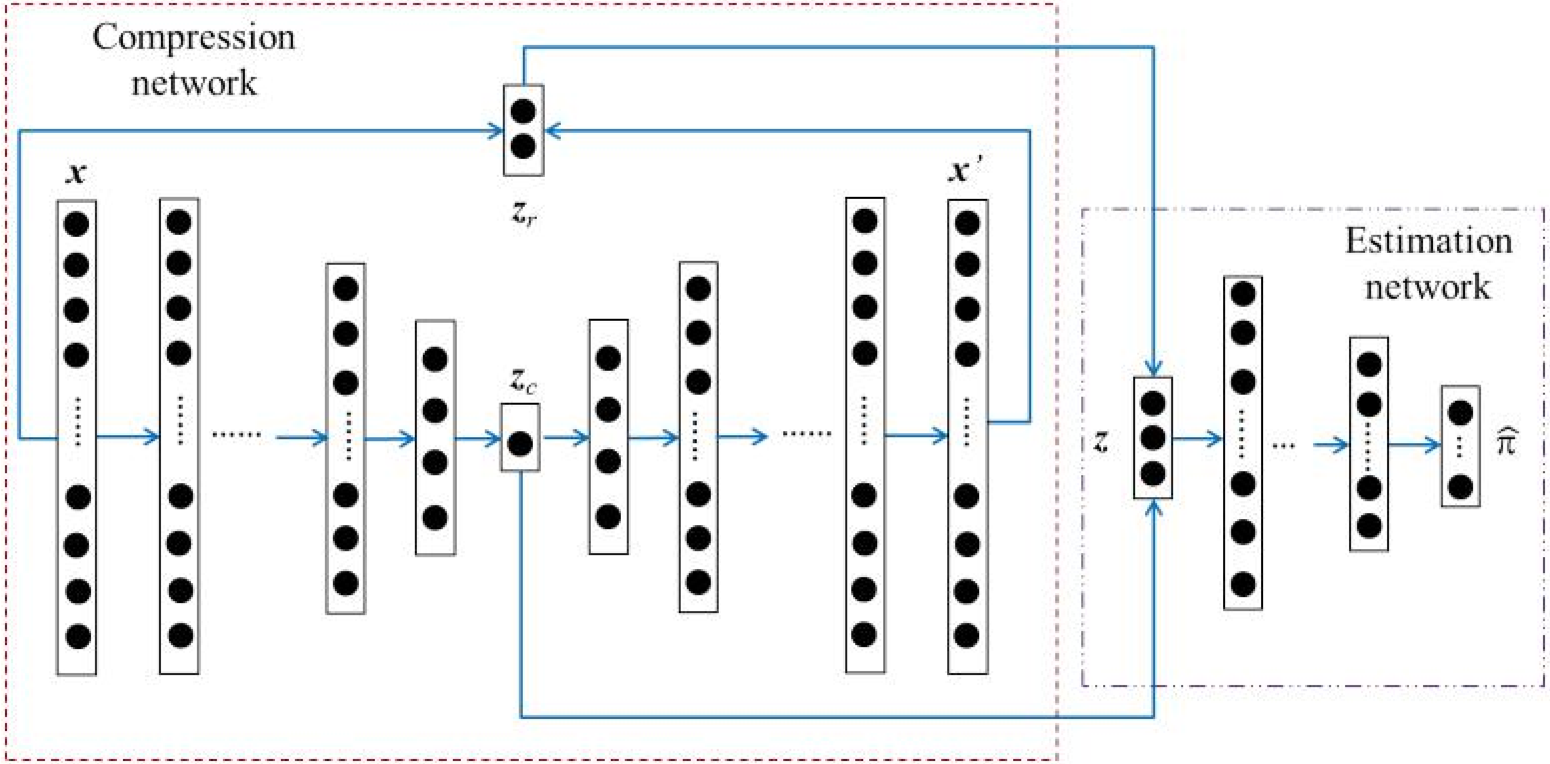


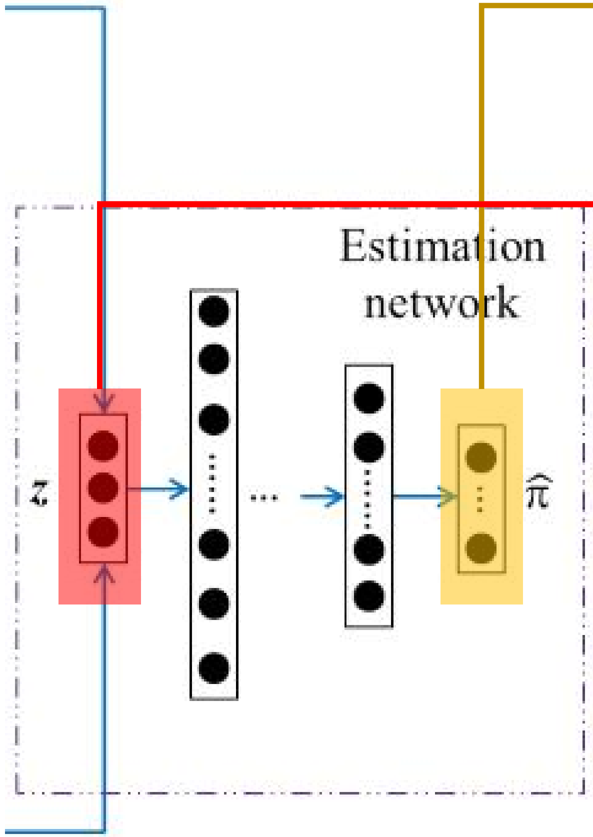

Figure 2: An overview on Deep Autoencoding Gaussian Mixture Model

$$\mathbf{z}_c = h(\mathbf{x}; \theta_e), \qquad \mathbf{x}' = g(\mathbf{z}_c; \theta_d),$$

$$\mathbf{z}_r = f(\mathbf{x}, \mathbf{x}'),$$

$$\mathbf{z} = [\mathbf{z}_c, \mathbf{z}_r],$$

# DEEP AUTOENCODING GAUSSIAN MIXTURE MODEL FOR UNSUPERVISED ANOMALY DETECTION




$$\mathbf{p} = MLN(\mathbf{z};\theta_m), \qquad \hat{\gamma} = \text{softmax}(\mathbf{p}),$$

$$\hat{\phi}_k = \sum_{i=1}^{N} \frac{\hat{\gamma}_{ik}}{N}, \qquad \hat{\mu}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} \mathbf{z}_i}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}, \qquad \hat{\mathbf{\Sigma}}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} (\mathbf{z}_i - \hat{\mu}_k)(\mathbf{z}_i - \hat{\mu}_k)^T}{\sum_{i=1}^{N} \hat{\gamma}_{ik}}$$

**GMM:**

$$\sum_{k=1}^{K} \hat{\phi}_k \frac{\exp\left(-\frac{1}{2}(\mathbf{z}-\hat{\mu}_k)^T \hat{\mathbf{\Sigma}}_k^{-1} (\mathbf{z}-\hat{\mu}_k)\right)}{\sqrt{|2\pi\hat{\mathbf{\Sigma}}_k|}}$$

**Sample energy:**

$$E(\mathbf{z}) = -\log\left(\sum_{k=1}^{K} \hat{\phi}_k \frac{\exp\left(-\frac{1}{2}(\mathbf{z}-\hat{\mu}_k)^T \hat{\mathbf{\Sigma}}_k^{-1} (\mathbf{z}-\hat{\mu}_k)\right)}{\sqrt{|2\pi\hat{\mathbf{\Sigma}}_k|}}\right)$$

# DEEP AUTOENCODING GAUSSIAN MIXTURE MODEL FOR UNSUPERVISED ANOMALY DETECTION

## Objective function:

$$J(\theta_e, \theta_d, \theta_m) = \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{x}'_i) + \frac{\lambda_1}{N}\sum_{i=1}^{N} E(\mathbf{z}_i) + \lambda_2 P(\hat{\mathbf{\Sigma}}).$$

$$L(\mathbf{x}_i, \mathbf{x}'_i) = \|\mathbf{x}_i - \mathbf{x}'_i\|_2^2.$$

$$P(\hat{\mathbf{\Sigma}}) = \sum_{k=1}^{K}\sum_{j=1}^{d} \frac{1}{\hat{\mathbf{\Sigma}}_{kjj}}$$

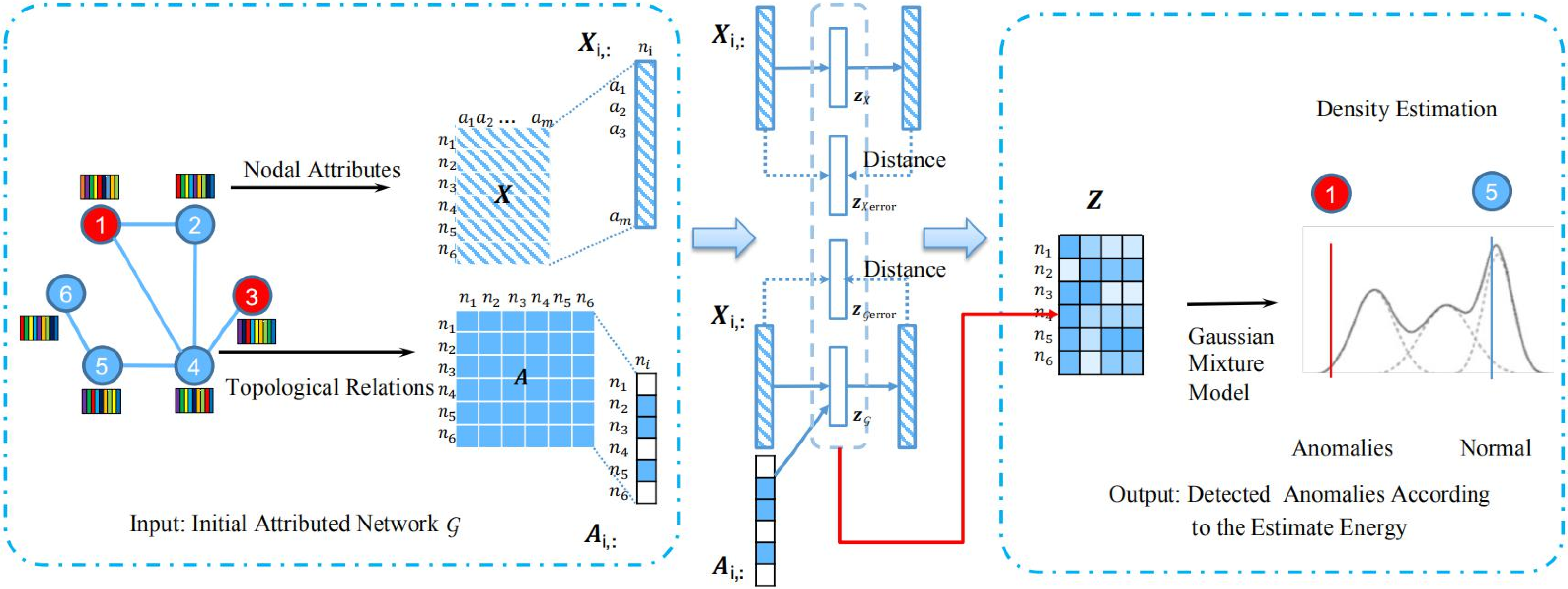
$$E(\mathbf{z}) = -\log\left(\sum_{k=1}^{K} \hat{\phi}_k \frac{\exp\left(-\frac{1}{2}(\mathbf{z} - \hat{\mu}_k)^T \hat{\mathbf{\Sigma}}_k^{-1}(\mathbf{z} - \hat{\mu}_k)\right)}{\sqrt{|2\pi\hat{\mathbf{\Sigma}}_k|}}\right)$$

# SpecAE: Spectral AutoEncoder for Anomaly Detection in Attributed Networks

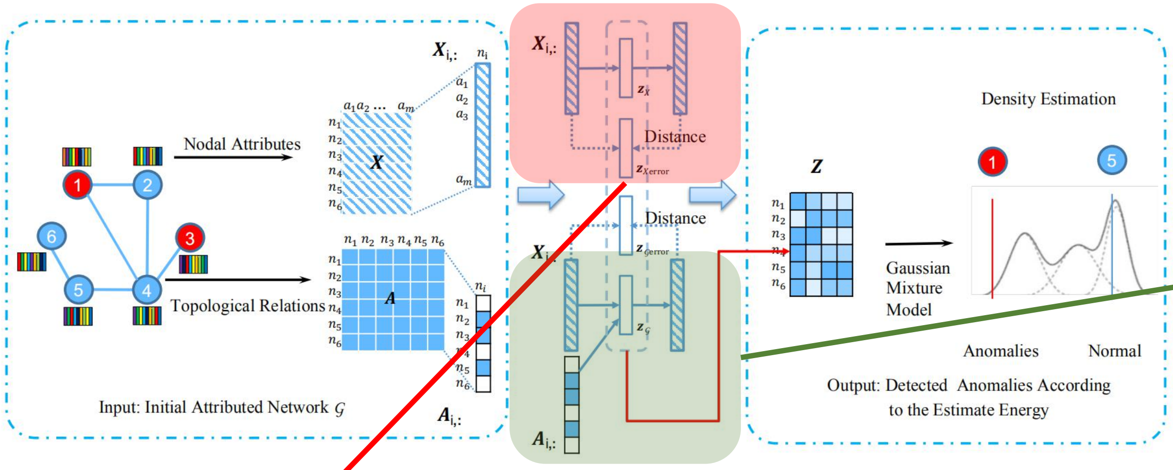
Yuening Li[1], Xiao Huang[1], Jundong Li[2,3], Mengnan Du[1], Na Zou[4]
[1]Department of Computer Science and Engineering, Texas A&M University
[2]Department of Electrical and Computer Engineering, University of Virginia
[3]Department of Computer Science & School of Data Science, University of Virginia
[4]Department of Industrial & Systems Engineering ,Texas A&M University
{yueningl,xhuang,dumengnan,nzou1}@tamu.edu,jl6qk@virginia.edu

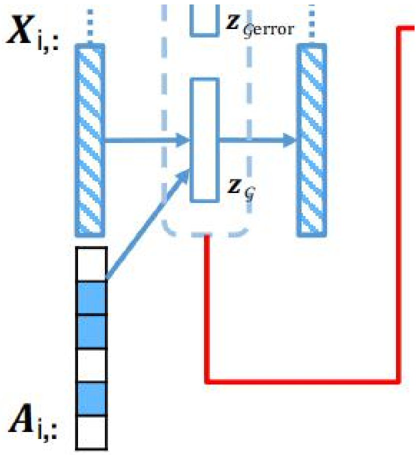
**CIKM-19**

# SpecAE: Spectral AutoEncoder for Anomaly Detection in Attributed Networks



# SpecAE: Spectral AutoEncoder for Anomaly Detection in Attributed Networks

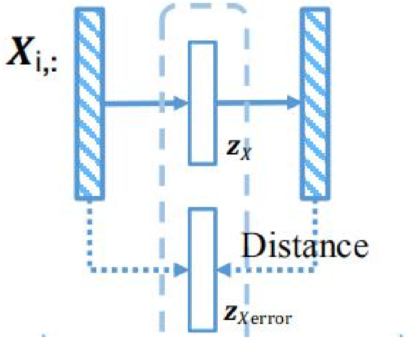



$$\mathrm{Conv}(X, A) = \sigma\left((1-\alpha)X + \alpha\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\right)W_f$$

$$\mathrm{Deconv}(Z, A) = \sigma\left((1+\alpha)Z - \alpha\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}Z\right)W_g$$

$$Z_{\mathcal{G}error} = \mathrm{dis}(X, \tilde{X})$$

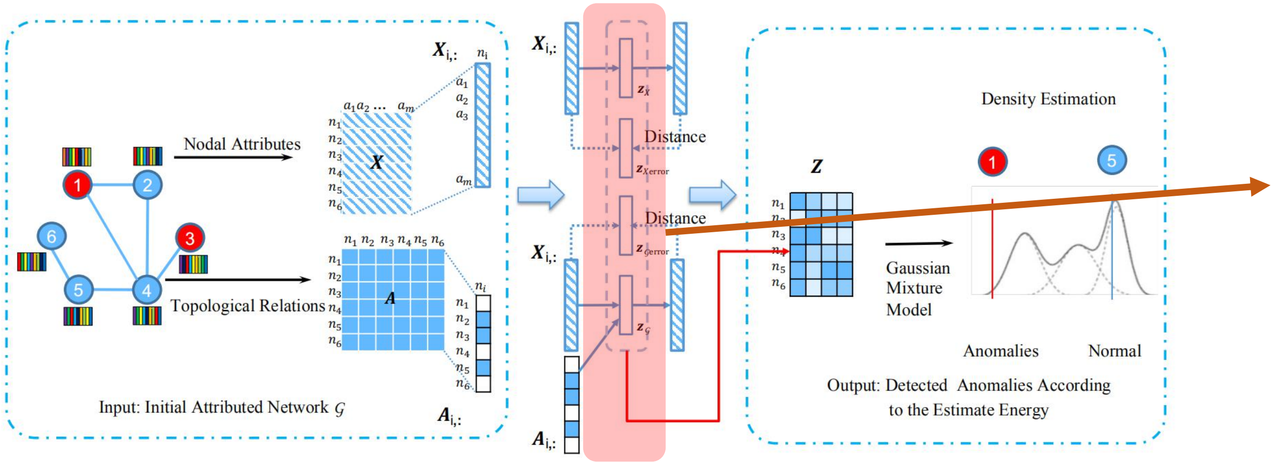


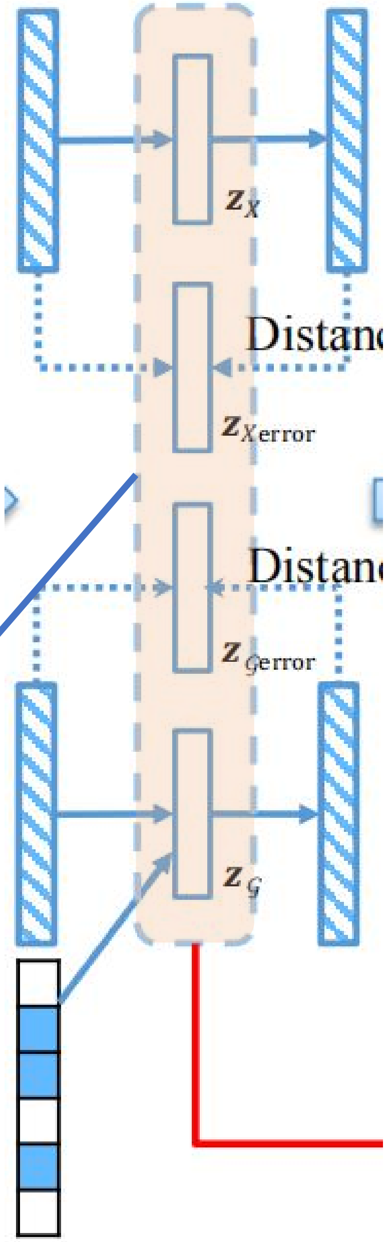

$$Z_X = \sigma(b_e + XW_e), \hat{X} = \sigma(b_d + Z_X W_d)$$

**Reconstruction errors:** $Z_{Xerror} = \mathrm{dis}(X, \hat{X})$

Euclidean distance and the cosine distance

# SpecAE: Spectral AutoEncoder for Anomaly Detection in Attributed Networks




$$f(\mathrm{Z}_{\mathcal{G}}|\mathrm{X},\mathrm{A}) = \prod_{i=1}^{n} f(\mathrm{Z}_{\mathcal{G}_{i,:}}|\mathrm{X}_{i,:},\mathrm{A}),$$

$$\text{with} \quad f(\mathrm{Z}_{\mathcal{G}_{i,:}}|\mathrm{X},\mathrm{A}) = \mathcal{N}(\mathrm{Z}_{\mathcal{G}_{i,:}}|\mu_{i,:},\mathrm{diag}(\sigma^2_{i,:}));$$

$$g(\widetilde{\mathrm{X}}|\mathrm{Z}_{\mathcal{G}},\mathrm{A}) = \prod_{i=1}^{n} g(\widetilde{\mathrm{X}}_{i,:}|\mathrm{Z}_{\mathcal{G}_{i,:}},\mathrm{A}),$$

$$\text{with} \quad g(\widetilde{\mathrm{X}}_{i,:}|\mathrm{Z}_{\mathcal{G}_{i,:}},\mathrm{A}) = \mathrm{Deconv}(\mathrm{Z}_{\mathcal{G}_{i,:}},\mathrm{A});$$

$$\mathrm{Z} = [\mathrm{Z}_X;\mathrm{Z}_{\mathcal{G}};\mathrm{Z}_{X\mathrm{error}};\mathrm{Z}_{\mathcal{G}\mathrm{error}}]$$

# SpecAE: Spectral AutoEncoder for Anomaly Detection in Attributed Networks

## The sample energy:

$$E(\mathbf{z}) = -\log\left(\sum_{k=1}^{K}\hat{\phi}_k \frac{\exp -\frac{1}{2}(\mathbf{z}-\hat{\mu}_k)^T\hat{\Sigma}_k^{-1}(\mathbf{z}-\hat{\mu}_k)}{\sqrt{|2\pi\hat{\Sigma}_k|}}\right)$$

**Gaussian Mixture Model (GMM)**

where we estimate the parameters in GMM as follows:

$$\hat{\gamma} = \text{softmax}(\mathbf{z}), \qquad \hat{\phi}_k = \sum_{i=1}^{N}\frac{\hat{\gamma}_{ik}}{N},$$

$$\hat{\mu}_k = \frac{\sum_{i=1}^{N}\hat{\gamma}_{ik}\mathbf{z}_i}{\sum_{i=1}^{N}\hat{\gamma}_{ik}}, \quad \hat{\Sigma}_k = \frac{\sum_{i=1}^{N}\hat{\gamma}_{ik}(\mathbf{z}_i-\hat{\mu}_k)(\mathbf{z}_i-\hat{\mu}_k)^T}{\sum_{i=1}^{N}\hat{\gamma}_{ik}}$$